Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πανεπιστήμιο Θεσσαλίας

Πολυ-συνεργασία στην ανίχνευση κίνησης και αναδρομολόγηση

Cooperative approach for traffic detection and rerouting

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Τάγκαλος

Επιβλέπων καθηγητής:

Δημήτριος Κατσαρός, Επίκουρος Καθηγητής

Δεύτερος Επιβλέπων καθηγητής:

Ελευθέριος Τσουκαλάς, Καθηγητής

Βόλος, Οκτώβριος 2017

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πανεπιστήμιο Θεσσαλίας

Πολυ-συνεργασία στην ανίχνευση κίνησης και αναδρομολόγηση

Cooperative approach for traffic detection and rerouting

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Τάγκαλος

Επιβλέπων Α’:
Δημήτριος Κατσαρός
Επίκουρος Καθηγητής

Επιβλέπων Β’:
Ελευθέριος Τσουκαλάς
Καθηγητής

Βόλος, Οκτώβριος 2017

Στην οικογένεια μου

Στην μνήμη της γιαγιάς μου που έφυγε πρόσφατα

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον κ. Δημήτριο Κατσαρό, ως επιβλέποντα καθηγητή της παρούσας διπλωματικής εργασίας, για τις χρήσιμες συμβουλές και την καθοδήγηση του, μέχρι την ολοκλήρωση της εργασίας. Επίσης θα ήθελα να τον ευχαριστήσω για τις καίριες ενέργειες που πραγματοποίησε έτσι ώστε να μπορέσει να ολοκληρωθεί και να παρουσιαστεί η παρούσα διπλωματική εργασία. Επιπλέον, θέλω να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Ελευθέριο Τσουκαλά για την πολύτιμη βοήθειά του.

Επίσης θα ήθελα να ευχαριστήσω τον συμφοιτητή μου Κωνσταντίνο Μανωλιό, για την πολύτιμη βοήθεια και στήριξη που μου παρείχε όλα αυτά τα χρόνια, στην κοινή μας πορείας για την ολοκλήρωση των σπουδών μας.

Ακόμη, ευχαριστώ θερμά τους φίλους μου, για όλα όσα μου πρόσφεραν όλα αυτά τα χρόνια και για όλες εκείνες τις στιγμές που ζήσαμε μαζί.

Τέλος το μεγαλύτερο ευχαριστώ, οφείλω να το δώσω στους γονείς μου, Αντώνη και Εύη, και στον αδελφό μου, Σπύρο. Τους ευχαριστώ για την αγάπη τους, την υπομονή τους και την στήριξη τους, σε κάθε επίπεδο, όλα αυτά τα χρόνια των σπουδών μου. Στάθηκαν δίπλα μου, βιώνοντας μαζί μου, τις χαρές και τις λύπες, τις αγωνίες και τις προσπάθειες, τις «νίκες» και τις «ήττες».

Σύνοψη

Στόχος της παρούσας διπλωματικής εργασίας, είναι να χρησιμοποιήσει τις δυνατότητες που προσφέρουν οι συσκευές smartphone και συγκεκριμένα το λειτουργικό σύστημα Android, για την ανάπτυξη μιας εφαρμογής για κινητές συσκευές. Η εφαρμογή που σχεδιάστηκε, έχει ως κύριο σκοπό να βοηθήσει τους χρήστες της να φτάσουν γρηγορότερα στον προορισμό τους. Για να το πετύχει αυτό, εκτός από την λειτουργία της εμφάνισης της προτεινόμενης διαδρομής, θα πρέπει να υπάρχει μια λειτουργία η οποία θα συλλέγει δεδομένα σχετικά με την κίνηση στους δρόμους και πιθανές καθυστερήσεις που μπορεί να υπάρχουν. Με αυτό τον τρόπο ο χρήστης, παίζει πολύ σημαντικό ρόλο στην εφαρμογή, λόγου του ότι θα πρέπει να συμβάλει στην συλλογή των δεδομένων για να μπορέσει να λειτουργεί σωστά ο αλγόριθμος.

Για τη σωστή λειτουργία της εφαρμογής, χρειάστηκε να δημιουργηθεί ένα σύστημα με διάφορες από διάφορες υπομονάδες όπως είναι ο server, η βάση δεδομένων και το Google API. Οι υπομονάδες αυτές, επικοινωνούν μεταξύ τους, παρέχοντας στην εφαρμογή όσο τον δυνατόν ακριβέστερα και καλύτερα αποτελέσματα. Να σημειωθεί ότι ο server,αποτελεί μια από τις σημαντικότερες υπομονάδες του συστήματος της εφαρμογής, αφού μέσω αυτού γίνεται δυνατή η επικοινωνία της βάσης δεδομένων και του Google API, όπου είναι αποθηκευμένες οι πληροφορίες για την κίνηση στους δρόμους. Επίσης στον server υπάρχει και ο αλγόριθμος μέσα από τον οποίον ανακαλύπτεται η γρηγορότερη διαδρομή.

Επιπλέον, για την εύρυθμη λειτουργία της εφαρμογής χρειάστηκε να δημιουργηθεί ο αλγόριθμος FastWay. Ο συγκεκριμένος αλγόριθμος λαμβάνει ως είσοδο την διαδρομή για την οποία ενδιαφέρεται ο χρήστης, δίνοντας του την δυνατότητα να ορίσει την μέρα και την ώρα που τον ενδιαφέρει. Έχοντας αυτά τα δεδομένα ως είσοδο, ψάχνει στην βάση δεδομένων να βρει τα κατάλληλα στοιχεία έτσι ώστε να τα συνδυάσει σωστά και να δώσει στον χρήστη την γρηγορότερη διαδρομή.

Τέλος να τονιστεί ότι η εφαρμογή δημιουργήθηκε με σκοπό να είναι όσο το δυνατόν πιο φιλική προς τον χρήστη, ενώ οι απαντήσεις που λαμβάνει από τον server είναι άμεσες, με πιο αργή να είναι αύτη της απάντησης του αλγορίθμου που χρειάζεται περίπου 1,5 δευτερόλεπτο για να λάβουμε την γρηγορότερη διαδρομή.

Abstract

The aim of this thesis project is the creation of a smartphone application exploiting the possibilities offered by smartphone devices and more specifically devices running under the Android OS. The purpose of the application is to assist users to reach their desired destination in a shorter time. To achieve this, two functionalities have to be offered, first a functionality that displays a recommended route, and secondly a functionality that collects data regarding the road traffic levels and possible delays visible on the road network. This way, the user plays a key role to the core functionality of the application as he has to contribute to the data gathering in order for the system to operate effectively.

In order for the functionality to be implemented, a whole system had to be created that consist of various combined subsystems including the server, database and the Google API. These subsystems co-operate to provide as good and as precise results as possible. The most critical part of the system is the server, which links the application to the database and Google API, where all traffic information is stored. Moreover, the Server runs the algorithm which discovers the best route.

For the core functionality of the application an algorithm called FastWay was built. The algorithm receives as an input the source and destination that the user is interested about, the time and date of interest, which are combined in order to return the result containing the fastest route.

The application was designed in such a way that it will be as user friendly as possible in terms of the user interface. Moreover, the results are received instantly, with the lowest response time being that of the algorithm which requires approximately 1,5 seconds to return the fastest route.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1.Σκοπός εργασίας

Αντικείμενο της παρούσας διπλωματικής εργασίας, αποτελεί η ανάπτυξη και η σχεδίαση μιας εφαρμογής για smartphones, σε λειτουργικό σύστημα Android. Βασική επιδίωξη της εφαρμογής αυτής, είναι η ανίχνευση της κίνησης στους δρόμους, με σκοπό να καταστεί εφικτή η παρουσίαση μιας εναλλακτικής διαδρομής στον χρήστη της εφαρμογής.

Για την επίτευξη του στόχου αυτού, η εφαρμογή χρειάζεται να ενσωματώνει δυο βασικές λειτουργίες. Αρχικά, η πρώτη λειτουργία, σχετίζεται με την διαδρομή η οποία θα προτείνεται στον χρήστη. Επιπλέον, η δεύτερη λειτουργία, αφορά την περισυλλογή πληροφοριών, οι οποίες σχετίζονται με την κίνηση στους δρόμους, με σκοπό να παρουσιάζεται στον χρήση η πιο σύντομη και «αποδοτικότερη» διαδρομή.

1.2. Δομή Εργασίας

Η διπλωματική εργασία αποτελείται από επτά βασικά κεφάλαια, με το καθένα να αποτελείται από διαφορετικές υποενότητες.

- Στο πρώτο κεφάλαιο επιχειρείται μια εισαγωγή στο πρόβλημα που καλείται να λύσει η συγκεκριμένη εφαρμογή καθώς επίσης παρουσιάζεται ο τρόπος με τον οποίο αντιμετωπίζεται.
- Στο δεύτερο κεφάλαιο πραγματοποιείται μια σύντομη εισαγωγή και ανάλυση του λογισμικού Android. Στην συνέχεια επιχειρείται μια ιστορική αναδρομή του λογισμικού καθώς και μια σύντομη ανάλυση για κάθε νέα έκδοση που έχει κυκλοφορήσει. Στο τέλος του κεφαλαίου συναντάμε την ανάλυση της αρχιτεκτονικής του και το λόγο για τον οποίο κάποιος χρησιμοποιεί το συγκεκριμένο λογισμικό.
- Στο τρίτο κεφάλαιο αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν στην διπλωματική εργασία, με βασικότερες να είναι αυτές των Web Services και του Google Maps Api.
- Στο τέταρτο κεφάλαιο αναλύονται τα λογισμικά, τα οποία είναι απαραίτητα για την δημιουργία του συστήματος της εργασίας.
- Στο πέμπτο κεφαλαίο παρουσιάζεται η αρχιτεκτονική της εφαρμογής. Αρχικά αναλύεται ο τρόπος με τον οποίο συνδέονται τα διαφορετικά τμήματα του συστήματος, ενώ στην συνέχεια περιγράφεται η λειτουργία που επιτελεί κάθε υπομονάδα.
- Στο έκτο κεφάλαιο γίνεται αναλυτική περιγραφή του τρόπου με τον οποίο λειτουργούν τα δύο βασικά μέρη της εργασίας.
- Στο έβδομο κεφάλαιο περιέχονται στοιχεία σχετικά με τις συσκευές που χρησιμοποιήθηκαν για τις δοκιμές της εφαρμογής και κάποια συμπεράσματα σχετικά με την διπλωματική εργασία.

Στη συνέχεια υπάρχει ένα παράτημα, το οποίο περιέχει όλο τον κώδικα που χρησιμοποιήθηκε τόσο από την εφαρμογή όσο και από τον server.

Στο τέλος παρατίθενται οι σχετικές βιβλιογραφικές αναφορές

Κεφάλαιο 2

Λειτουργικό Android

Περιεχόμενα Κεφαλαίου

2.1.Λειτουργικό σύστημα Android

Το Android είναι ένα λειτουργικό σύστημα, το οποίο συναντάται σε ένα εύρος ηλεκτρονικών συσκευών, με τις πιο διαδεδομένες αυτές των smartphones. Μια πληθώρα κινητών smartphones, χρησιμοποιούν λογισμικό Android, με εξαίρεση τις συσκευές της Apple, που χρησιμοποιούν IOS, και αυτές της Microsoft, που χρησιμοποιούν Windows. Επιπλέον το λογισμικό αυτό συναντάται συχνά σε tablet, τηλεοράσεις και άλλα gadget.

Εικόνα 2.1 Λογότυπο Android

Το λειτουργικό σύστημα Android δημιουργήθηκε από την Google και έχει ως βάση τον πυρήνα των Linux. Όπως και ο πηγαίος κώδικας των Linux έτσι και αυτός των Android, είναι ανοιχτός και ελεύθερος προς ανάπτυξη και παραμετροποίηση από υποψήφιους προγραμματιστές. Έτσι πολλές εταιρίες οι οποίες κατασκευάζουν κινητά τηλέφωνα όπως είναι η HTC, η Samsung και η Huawei έχουν την δυνατότητα να παίρνουν αυτούσιο τον κώδικα και να τον επεξεργάζονται κατά το δοκούν προκυμμένου να μπορούν να το τρέξουν στις δικές τους κινητές συσκευές που θα παρουσιάσουν στο αγοραστικό κοινό.

Γενικά το λειτουργικό σύστημα Android, βοηθάει το χρήστη να εκμεταλλευτεί όλες αυτές τις δυνατότητες που του δίνει το hardware της συσκευής του. Σε αυτό μπορούμε να εγκαταστήσουμε μια μεγάλη γκάμα εφαρμογών, οι οποίες εκμεταλλεύονται το hardware της εκάστοτε συσκευής, πολλές εκ των οποίων έχουν δημιουργηθεί από την ίδια την Google. Η δημιουργία τέτοιων εφαρμογών, έχει ως αποτέλεσμα τη διευκόλυνση της καθημερινότητας των χρηστών smartphones, αφού τους δίνεται η δυνατότητα να αναζητήσουν διάφορες πληροφορίες στο διαδίκτυο, να ακούσουν μουσική, να δουν βίντεο, να εντοπίσουν την τοποθεσία τους στον χάρτη, να βγάλουν φωτογραφίες με την κάμερα του τηλεφώνου και πολλές ακόμα λειτουργίες. Ένα ακόμα πλεονέκτημα του λογισμικού αυτού, αποτελεί η δυνατότητα προσαρμογής της εμφάνισης του με βάση τις επιθυμίες του εκάστοτε χρήστη. Ποιο συγκεκριμένα, αυτό επιτυγχάνεται μέσω της χρήσης wallpapers, θεμάτων και launchers, τα οποία μπορεί να κάνουν το interface του κάθε κινητού διαφορετικό και ξεχωριστό.

Εκτός όμως από τις βασικές εφαρμογές οι οποίες υπάρχουν εγκατεστημένες στο κινητό τηλέφωνο και εκμεταλλεύονται το hardware, υπάρχουν και αυτές που μπορεί να κατεβάσει ο χρήστης, οι οποίες του δίνουν ένα μεγάλο εύρος δυνατοτήτων. Κλασσικό παράδειγμα αυτών των εφαρμογών, αποτελεί το Facebook, αλλά και άλλες επιπλέον οι οποίες βοηθούν στις τραπεζικές σου συναλλαγές ή στο να παραγγείλεις φαγητό.

Συνοψίζοντας αντιλαμβανόμαστε ότι το λογισμικό android επηρεάζει σε σημαντικό βαθμό την καθημερινότητα μας, καθώς έχει μετατρέψει τα κινητά τηλεφωνά σε έναν προσωπικό υπολογιστή τσέπης, καθιστώντας τον προσβάσιμο κάθε στιγμή της ημέρας.

2.2.Ιστορία Android

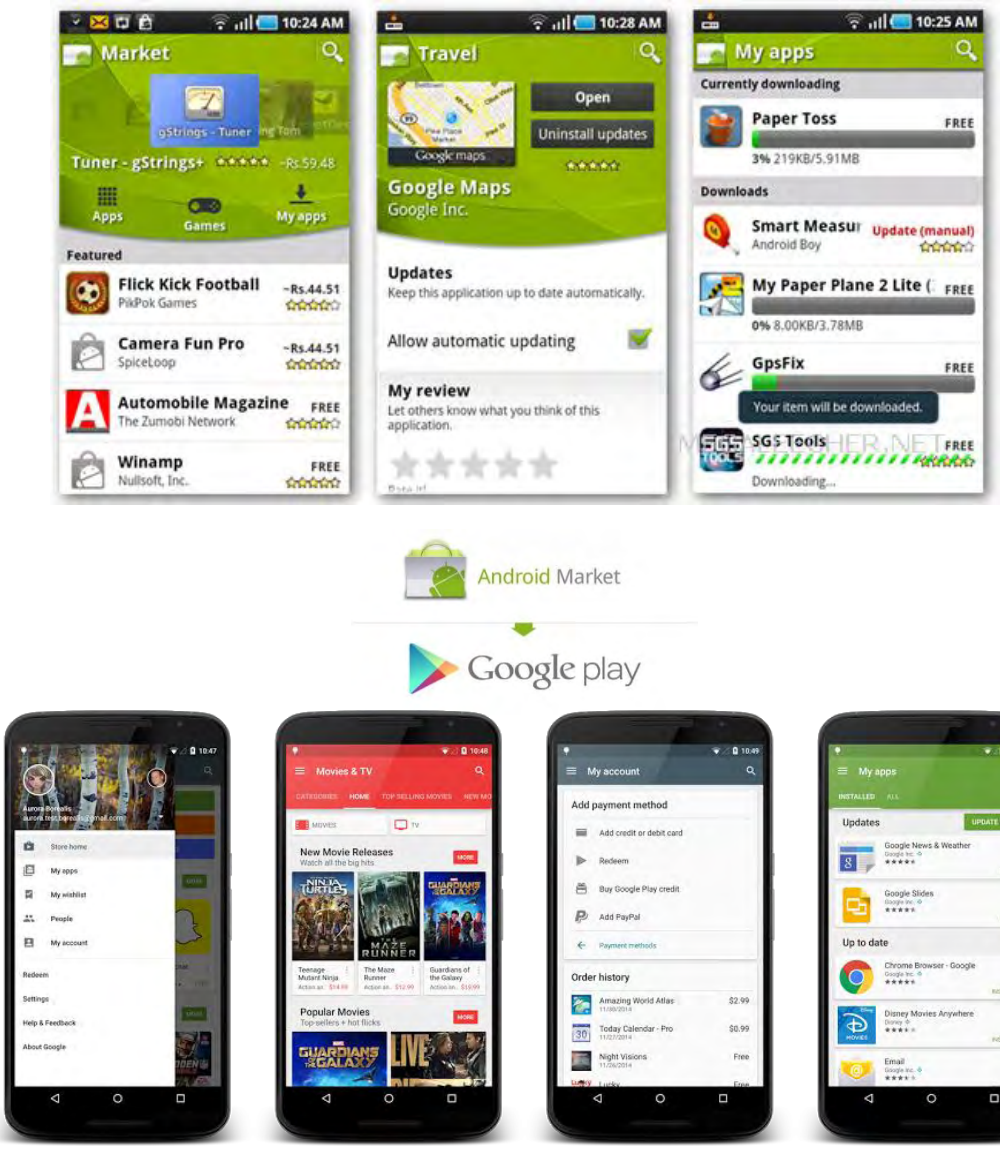
Τον Οκτώβριο του 2003, στο Palto Alto της Califronia, ιδρύθηκε η εταιρεία Android Inc από τους Rich Miner, Nick Sears, Chris White και Andy Rubin. Την ημέρα της ίδρυσης της ένας, εκ των ιδρυτών, ο Rubin, δήλωσε σχετικά με την νεοσυσταθείσα εταιρεία ότι επρόκειτο να αναπτύξουνε “τις ευφυέστερες κινητές συσκευές που θα έχουν μεγαλύτερη επίγνωση της θέσης και των προτιμήσεων του κατόχου τους”.

Ο πρωταρχικός στόχος της Android Inc, ήταν η ανάπτυξη ενός λογισμικού συστήματος για φωτογραφικές μηχανές. Ένα τέτοιο λογισμικό, παρουσιάστηκε σε επενδυτές το 2004, όπου το λειτουργικό Android, είχε εγκατασταθεί σε μια ψηφιακή μηχανή η οποία συνδεόταν ασύρματα με έναν υπολογιστή. Αυτός ο υπολογιστής θα συνδεόταν στη συνέχεια σε έναν “Android DataCenter”, όπου οι κάτοχοι φωτογραφικών μηχανών θα μπορούσαν να αποθηκεύουν τις φωτογραφίες τους online σε έναν Cloud Server. Ωστόσο έπειτα από έρευνες που διεξήχθησαν, κατέληξαν στο συμπέρασμα ότι η αγορά δεν θα μπορούσε να ανταποκριθεί σε ένα τέτοιο λογισμικό και στράφηκαν στην δημιουργία λογισμικού για smartphones.

Τον Ιούλιο του 2005 η Android Inc εξαγοράστηκε από την Google, για ένα πόσο που δεν έχει γίνει ακόμα γνωστό, αλλά ήταν τουλάχιστον της τάξεως των 50 εκατομμυρίων δολαρίων. Ο Rubin και οι άλλοι συνιδρυτές, συνέχισαν να δουλεύουν πάνω στην ανάπτυξη του λειτουργικού, αύτη την φορά όμως κάτω από την νέα ιδιοκτησιακή στέγη της Android Inc, με τον Rubin μάλιστα να αποτελεί τον υπεύθυνος της ομάδας Android μέχρι και το 2013. Με την αγορά της Android Inc από την Google, λήφθηκε η απόφαση ότι η βάση του λογισμικού αυτού θα είναι ο πυρήνας τον Linux, με αποτέλεσμα να διατίθεται ελεύθερα σε κατασκευαστές κινητών τηλεφώνων. Η Google θεώρησε ότι αναπτύσσοντας άλλες υπηρεσίες που χρησιμοποιούνται από το λογισμικό σύστημα αλλά και από άλλες εφαρμογές θα της επέφερε περισσότερα κέρδη.

Το 2007 αποτελεί κομβική χρονιά στον τομέα των smartphones, με την Apple να κάνει την αρχή θέτοντας σε κυκλοφορία το πρώτο της Iphone. Η Google έπρεπε να εντείνει τους ρυθμούς ανάπτυξης του νέου της λογισμικού, έτσι ώστε να παρουσιάσει την πρώτη έκδοση του. Έτσι, στις 5 Νοεμβρίου του 2007, έχουμε την πρώτη παρουσίαση της πλατφόρμας Android. Την ίδια χρονιά, ανακοινώνεται και η ίδρυση ενός οργανισμού ο οποίος συμπεριλάμβανε ένα μεγάλο αριθμό τηλεπικοινωνιακών εταιριών, που ασχολούνται τόσο με την ανάπτυξη λογισμικού όσο και την κατασκευή hardware. Ορισμένες από τις εταιρίες ήταν η Google, η T-Mobile, η Motorola, η Samsung , η Qualcomm, η Texas Instruments και άλλες. Το όνομα του οργανισμού αυτού, είναι Open Handset Alliance, και στοχεύει στη δημιουργία ενός λογισμικού ελεύθερου και ανοιχτού προς το κοινό. Αυτό δίνει τη δυνατότητα σε άτομα που διαθέτουν προγραμματιστικές ικανότητες, είτε να δημιουργήσουν τη δική τους έκδοση λογισμικού είτε να αναπτύξουν τη δική τους εφαρμογή για Android smartphones. Για το λόγο αυτό, το μεγαλύτερο κομμάτι του κώδικα του Android, δημοσιεύτηκε από την Google υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού, η οποία παρέχει την δυνατότητα σε πλήθος προγραμματιστών να σχεδιάσει και να αναπτύξει τις δίκες του εφαρμογές. Επιπλέον, το πρώτο κινητό στο οποίο χρησιμοποιήθηκε λογισμικό Android ήταν το HTC Dream το οποίο κυκλοφόρησε στην αγορά το 2008. Τέλος να σημειωθεί ότι στις 28 Αυγούστου 2008, η Google ανακοινώσε το Android Market(όπου από τις 6 Μαρτιού 2012 ονομάζεται Google Play), το οποίο πλέον αποτελεί το επίσημο κατάστημα για τις εφαρμογές Android. Μέσω αυτού, δίνεται η δυνατότητα στους χρήστες να κατεβάζουν εφαρμογές στη συσκευή τους,είτε ελεύθερα είτε επί πληρωμή.

Εικόνα 2.2 HTC Dream



Εικόνα 2.3 Android Market σε Google Play

2.3. Εκδόσεις Android

Μετά την ανακοίνωση της πρώτης έκδοσης Android, κυκλοφόρησαν πολλές αναβαθμίσεις, η καθεμία από τις οποίες πρόσθετε νέα χαρακτηριστικά και λειτουργίες. Αξιοσημείωτο μπορεί να χαρακτηριστεί το γεγονός ότι κάθε νέα αναβάθμιση έπαιρνε το όνομα της από κάποιο γλύκισμα και μάλιστα διατηρήθηκε αλφαβητική σειρά στην ονοματολογία. Έως τώρα στην αγορά έχουν κυκλοφορήσει 15 εκδόσεις οι οποίες θα παρουσιαστούν παρακάτω και θα αναφερθούν κάποια βασικά χαρακτηριστικά τους.

Όπως αναφέραμε και παραπάνω, η πρώτη συσκευή Android που κυκλοφόρησε στην αγορά ήταν το HTC Dream (G1), με Android 1.0 και API level 1. Έπειτα, στις 9 Φεβρουάριου 2009, παρουσιάστηκε το Android 1.1 και API Level 2, το οποίο αποτελούσε αναβάθμιση της προηγούμενης έκδοσης. Οι δύο αυτές εκδόσεις δεν υποστήριζαν την χρήση κουμπιών αφής.

Εικόνα 2.4 Εκδόσεις Android

2.4.1. Android 1.5 Cupcake

Το Android 1.5 Cupcake, είναι η πρώτη σημαντική αναβάθμιση, η οποία έχει το δικό της όνομα και κυκλοφόρησε επίσημα στις 27 Απριλίου 2009. Σε αυτή προστίθενται πολλές νέες λειτουργίες, με σημαντικότερες αυτές της υποστήριξης κουμπιών αφής και της περιστροφής οθόνης για οριζόντια και κάθετη χρήση του κινητού τηλεφώνου. Ακόμα, δόθηκε η δυνατότητα υποστήριξης widget και φακέλων στην αρχική οθόνη.

Εικόνα 2.5 Android Cupcake Mascot

2.4.2. Android 1.6 Donut

Το Android 1.6 Donut, είναι μια αναβάθμιση η όποια περιλαμβάνει λιγότερες βελτιώσεις σε σχέση με τον προκάτοχο της. Κυκλοφόρησε στις 15 Σεπτεμβρίου 2009 και έχει πιο γρήγορες ταχύτητες. Επιπλέον υποστηρίζονται οθόνες μεγαλύτερων αναλύσεων και σημαντική προσθήκη αποτελούν οι χάρτες της Google.

Εικόνα 2.6 Android Donut Mascot

2.4.3. Android 2.0-2.1 Éclair

Η έκδοση Android 2.0, κυκλοφόρησε στις 26 Οκτωβρίου 2009. Λίγους μήνες αργότερα και συγκεκριμένα στις 12 Ιανουαρίου 2010, κυκλοφόρησε η έκδοση 2.1 . Η έκδοση 2.0 χρησιμοποιεί API 5 και η έκδοση 2.1, API 6. Βασικά χαρακτηριστικά αυτών των εκδόσεων αποτελούν, η γρηγορότερη απόκριση σε σχέση με τις προηγούμενες εκδόσεις, η εισαγωγή ενός καινούργιου browser, ο οποίος υποστήριζε το πρότυπο HTML5, η βελτίωση των Google Maps και η δυνατότητα χρήσης πολλαπλών email σε μια συσκευή. Τέλος, η έκδοση Éclair, είναι πρώτη έκδοση που χρησιμοποιεί τη δυνατότητα Text to Speech.

Εικόνα 2.7 Android Éclair Mascot

2.4.4. Android 2.2 Froyo

Η έκδοση Android 2.2 Froyo, εμφανίζεται στις 20 Μαΐου 2010 και το όνομα της αποτελεί συντομογραφία του «frozen yogurt». Οι συσκευές που διέθεταν την έκδοση αυτή, μπορούσαν να εκμεταλλευτούν κάποιες νέες δυνατότητες, συμπεριλαμβανομένων αυτών του USB tethering και Wi-Fi hotspot. Επιπρόσθετος υπάρχει σημαντική βελτίωση στην ταχύτητα και την μνήμη. Ενσωματώνεται, επιπλέον, η V8 JavaScript engine του Chrome στον περιηγητή του Android.

Εικόνα 2.8 Android Froyo Mascot

2.4.5. Android 2.3 Gingerbread

Η έκδοση Android 2.3 Gingerbread, κυκλοφόρησε στις 6 Δεκεμβρίου 2010. Αποτελεί τη μοναδική έκδοση από τις προαναφερθείσες, όπου ένα πολύ μικρό ποσοστό των χρηστών Android (0,6%), συνεχίζει να τη χρησιμοποιεί ακόμα και σήμερα. Σε αύτη την έκδοση παρατηρείται μια σημαντική ανανέωση στο user interface, ένα ανανεωμένο εικονικό πληκτρολόγιο καθώς και η δυνατότητα αντιγραφής-επικόλλησης. Επιπλέον, αρχίζουν να υποστηρίζονται περισσότερες από μια κάμερες για όσα τηλέφωνα διαθέτουν μπροστινή κάμερα. Τέλος, εμφανίζεται για πρώτη φορά η υπηρεσία Near Field Communication(NFC), επιτρέποντας στον χρήστη να διαβάσει μια NFC ετικέτα ενσωματωμένη σε ένα πόστερ, αυτοκόλλητο ή διαφημιστικό.

Εικόνα 2.9 Android Gingerbread Mascot

2.4.6. Android 3.0 Honeycomb

Η έκδοση Android 3.0 Honeycomb, κυκλοφόρησε στις 22 Φεβρουαρίου 2011. Η συγκεκριμένη έκδοση διαφοροποιείται από τις προηγούμενες, καθώς είναι η μοναδική αναβάθμιση αποκλειστικά για tablet. Απόρροια της ιδιαιτερότητας αυτής, αποτελεί ο επανασχεδιασμός του User Interface προκειμένου να προσαρμοστεί στις μεγάλες οθόνες των tablet. Πλέον ενσωματώνονται εικονικά κουμπιά στο κάτω μέρος της οθόνης και προστίθεται η Action Bar στο πάνω μέρος της.

Εικόνα 2.10 Android Honeycomb Mascot

2.4.7. Android 4.0 Ice Cream Sandwich

Η έκδοση Android 4.0 Ice Cream Sandwich, έγινε διαθέσιμη στο κοινό στις 18 Οκτωβρίου 2011. Αποτελεί μια από τις σημαντικότερες εκδόσεις της εταιρίας, καθώς συνδυάζει τις καλύτερες λειτουργίες των δυο προηγούμενων εκδόσεων, της Gingerbread και της Honeycomb. Ο χρήστης έχει τη δυνατότητα να τερματίζει τις εφαρμογές οι οποίες λειτουργούν στο παρασκήνιο, να θέτει όρια στην κίνηση δεδομένων και να αποκτά άμεση πρόσβαση στις εφαρμογές από την οθόνη κλειδώματος. Τέλος, η συσκευή ξεκλειδώνεται με μια πρωτοποριακή μέθοδο αναγνώρισης προσώπου του χρήστη.

Εικόνα 2.11 Android Ice Cream Sandwich Mascot

2.4.8. Android 4.1-4.3 Jelly Bean

Η εποχή του ονόματος Jelly Bean ξεκινάει στις 9 Ιουλίου 2012 με την έκδοση 4.1. Το ίδιο όνομα γλυκίσματος χρησιμοποιήθηκε σε δύο επιπλέον εκδόσεις, τις 4.2 και 4.3. Σημαντικές τροποποιήσεις παρατηρούνται στην μπάρα ειδοποιήσεων, με νέο περιερχόμενο και νέες επιλογές. Βελτίωση παρατηρείται στο πληκτρολόγιο, με την εισαγωγή της πρόβλεψης της επόμενης λέξης που εκτιμάται ότι θα χρησιμοποιήσει ο χρήστης και στη διαχείριση της ενέργειας όταν το κινητό βρίσκεται σε αδράνεια. Παράλληλα επιτρέπονται περισσότεροι λογαριασμοί χρηστών στην έκδοση που αφορά τα tablet.

Εικόνα 2.12 Android Jelly Bean Mascot

2.4.9. Android 4.4 KitKat

Το Android 4.4 είναι η πρώτη έκδοση λογισμικού η οποία λαμβάνει το όνομα της από κάποιο γλύκισμα το οποίο το όνομα του είναι ήδη trademarked. Για το λόγο αυτό η Google θα χρειαστεί να συνάψει συμφωνία με την Nestle για να βαφτίσει την έκδοση 4.4 με το όνομα KitKat. Αυτή η αναβάθμιση βγαίνει στην κυκλοφορία στις 31 Οκτωβρίου 2013. Σε αυτήν δεν υπάρχουν πολλά νέα χαρακτηριστικά, αυτό όμως που την κάνει να ξεχωρίσει από τις προηγούμενες είναι η

Εικόνα 2.13 Android KitKat Mascot

χαμηλότερη χρήση της μνήμης RAM, κάνοντάς το να είναι πιο ελαφρύ σαν λογισμικό. Επιπλέον συναντάμε ένα νέο γραφικό περιβάλλον αρκετά προσαρμοσμένο στο να διευκολύνει τον χρήστη.

Η μεγάλη διαφοροποίηση παρατηρείται στο status και στα notifications τα όποια πλέον είναι διάφανα και δεν είναι ορατά μέσα από τις εφαρμογές. Αναβαθμίσεις υπάρχουν και στις εφαρμογές κλήσεων και μηνυμάτων.

2.4.10. Android 5.0 Lollipop

Στις 12 Νοεμβρίου 2014 κυκλοφόρησε η έκδοση Android 5.0 Lollipop, η όποια άλλαξε ριζικά την εξωτερική εμφάνιση του λογισμικού και την έκανε γνωστή όπως είναι σήμερα. Η Google προτείνει ένα νέο σύνολο σχεδιαστικών κανόνων με όνομα Material Design. Το καινούργιο UI βασίζεται πάνω σε αυτό και είναι τελείως διαφορετικό με τα προηγούμενα, κάνοντας το παράλληλα και πιο εύχρηστο στο χρήστη. Επιπλέον αλλαγές υπάρχουν στα notifications, πλέον ο χρήστης μπορεί να καθορίσει από ποιους μπορεί να λαμβάνει ειδοποιήσεις. Επιπλέον η Google προσπαθεί να βοηθήσει το χρήστη να εξοικονομεί ενέργεια στην συσκευή του γι'αυτο εισάγει μια νέα λειτουργία την battery saver feature, που άμα επιλεγεί από κάποιον παρατείνει την μπαταρία του κατά 90 λεπτά. Εκτός όμως από τις οπτικές αλλαγές υπάρχουν και κάποιες αλλαγές που δεν είναι εμφανείς αλλά εκμεταλλεύονται το hardware των νέων smartphone, τέτοιες είναι η υποστήριξη πάνω από 1 SIM, οι κλήσεις υψηλής ευκρίνειας και η προστασία της συσκευής άμα κλαπεί ακόμα και άμα γίνει επαναφορά εργοστασιακών ρυθμίσεων.

Εικόνα 2.14 Android Lollipop Mascot

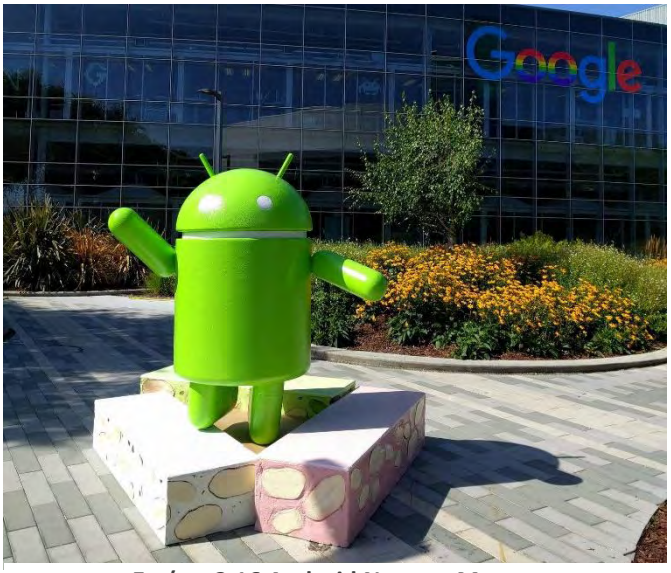
2.4.11. Android 6.0 Marshmallow

Η έκδοση Android 6.0 Marshmallow κυκλοφόρησε στις 5 Οκτωβρίου του 2015. Η σημαντικότερη βελτίωση παρατηρείται στο θέμα της ενέργειας, καθώς πλέον η συσκευή καταλάβει άμα χρησιμοποιείται ή όχι και αναλόγως προσαρμόζει την φωτεινότητα της οθόνης. Επιπλέον γίνεται η πρώτη έκδοση η οποία υποστηρίζει την δυνατότητα αναγνώρισης δακτυλικών αποτυπωμάτων με αισθητήρες αφής, για το ξεκλείδωμα της συσκευής σου. Εισάγεται επιπλέον και το Android Pay το οποίο είναι ένα νέο σύστημα πληρωμών. Τέλος υπάρχει καλύτερη διαχείριση των δικαιωμάτων που θέλει να δώσει ο χρήστης στις εφαρμογές που χρησιμοποιεί.

Εικόνα 2.15 Android Marshmallow Mascot

2.4.12. Android 7.0 Nougat

Η έκδοση Android 7.0 Nougat είναι μια από τις τελευταίες αναβαθμίσεις του λειτουργικού και η εκδόθηκε στις 22 Αυγούστου 2016. Μια από τις σημαντικότερες λειτουργίες που εμφανίζονται σε



Εικόνα 2.16 Android Nougat Mascot

αυτό το λογισμικό είναι το multi-window view το οποίο επιτρέπει στον χρήστη να έχει ανοιχτές στην οθόνη του κινητού του δυο εφαρμογές ταυτόχρονα, για παράδειγμα να βλέπει ένα βίντεο

στο YouTube και παράλληλα να γράφει κάποιο μήνυμα στο Messenger. Συνεχίζει να προσπαθεί να βελτιωθεί και στο θέμα της ενέργειας της μπαταρίας γι' αυτό εισάγει το Doze on the Go, το οποίο είναι μια λειτουργία που βοηθάει στην εξοικονόμηση όταν ο χρήστης βρίσκεται σε κίνηση και δεν χρησιμοποιεί το κινητό. Διαφορές υπάρχουν και στην notification bar σε σχέση με τις προηγούμενες εκδόσεις καθώς εισάγεται η επιλογή της άμεσης απάντησης. Έτσι μπορεί να απαντήσει ο χρήστης σε κάποιο μήνυμα απευθείας χωρίς να χρειάζεται να ανοίξει την εφαρμογής, ακόμα εμφανίζεται και η επιλογή της πολλαπλής διαγραφής ειδοποιήσεων. Τέλος άξιο αναφοράς είναι ότι πλέον υποστηρίζεται και το Vulkan API το όποιο αφορά την καλύτερη απόδοση των 3D γραφικών.

2.4.13. Android 8.0 Oreo

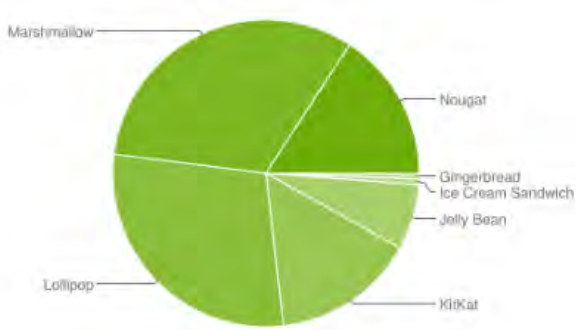
Το Android 8.0 Oreo είναι η πιο πρόσφατη έκδοση του λογισμικού και βγήκε στην κυκλοφορία στις 21 Αυγούστου 2017. Στην νέα αύτη έκδοση υπάρχει σημαντική βελτίωση στην ταχύτητα τόσο της εκκίνησης των εφαρμογών όσο και της ίδιας της συσκευή. Δίνεται η δυνατότητα στο χρήστη να διαλέξει ποια username και passwords θέλει κρατήσει στην μνήμη του έτσι ώστε την επόμενη φορά που πάει να το πληκτρολογήσει να το εμφανίσει αμέσως με την λειτουργία του Autofill. Τέλος υπάρχουν οπτικές αλλαγές στο μενού των ρυθμίσεων, όπως και η δυνατότητα picture-in-picture που επιτρέπει δυο εφαρμογές να λειτουργούν σαν μια.

Εικόνα 2.16 Android Oreo Mascot

2.4.14. Χρήση εκδόσεών Android

Σύμφωνα με τα στατιστικά στοιχεία της χρήσης των εκδόσεων που έδωσε η Google στην δημοσιότητα για την συλλογή δεδομένων κατά τις 7 πρώτες μέρες του Σεπτεμβρίου του 2017 παρατηρείται ότι η πιο δημοφιλής έκδοση είναι η Marshmallow.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.4%
4.2.x		17	3.5%
4.3		18	1.0%
4.4	KitKat	19	15.1%
5.0	Lollipop	21	7.1%
5.1		22	21.7%
6.0	Marshmallow	23	32.2%
7.0	Nougat	24	14.2%
7.1		25	1.6%

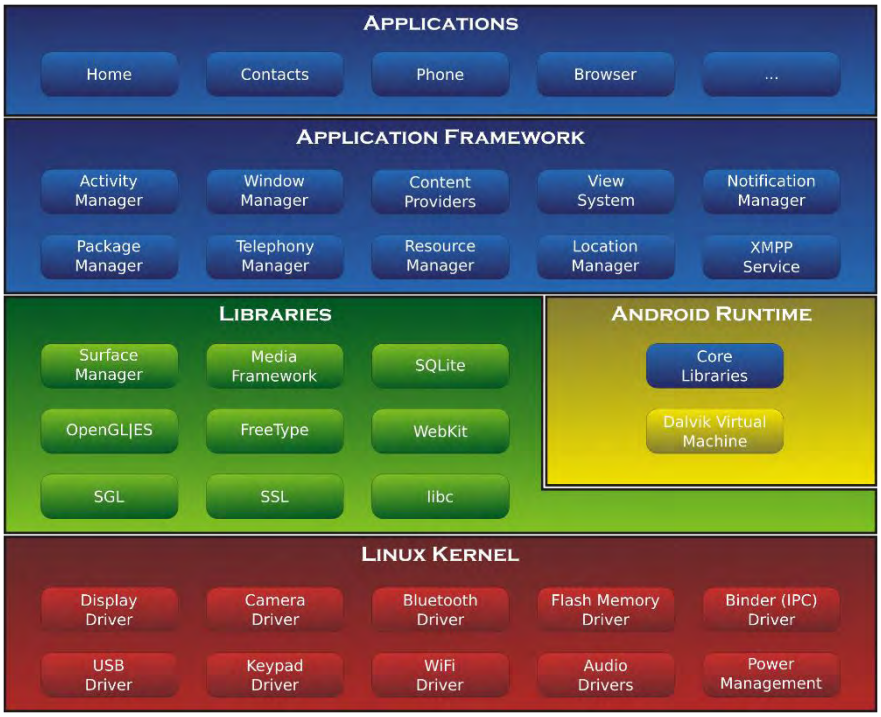


Εικόνα 2.17 Στατιστικά στοιχεία χρήσης εκδόσεων Android

2.4. Αρχιτεκτονική Android

Το λειτουργικό android είναι μια στοίβα από λογισμικά τα οποία εξυπηρετούν διαφορετικές λειτουργιές του συστήματος, η δομή αυτή είναι βασισμένη στην αρχιτεκτονική των Linux. Αποτελείται από 5 στρώματα εκ των οποίων τα 4 είναι τα σημαντικότερα . Τα στρώματα αυτά από το χαμηλότερο στο υψηλότερο είναι :

- Πυρήνας (Kernel) Linux
- Επίπεδο Βιβλιοθηκών (Libraries)
- Επίπεδο Εκτέλεσης (Android Runtime)
- Επίπεδο Πλαισίου Εφαρμογής (Application Framework)
- Επίπεδο Εφαρμογών (Applications)



Εικόνα 2.18 Αρχιτεκτονική Linux

2.4.1. Πυρήνας (Kernel) Linux

Ο πυρήνας Linux αποτελεί το χαμηλότερο επίπεδο αυτής της στοίβας λογισμικών. Ο πυρήνας λειτουργεί ως ένα ενδιάμεσο επίπεδο αφαίρεσης μεταξύ του λογισμικού και του hardware της συσκευής καθώς παρέχει όλους τους βασικούς οδηγούς υλικού για την σωστή λειτουργία τους, όπως της κάμερας, του πληκτρολογίου, της οθόνης κλπ. Επίσης επιτελεί κάποιες βασικές λειτουργίες, όπως της ασφάλειας και της διαχείρισης της μνήμης, για να διευκολύνει την ομαλή λειτουργία των εξωτερικών στοιχείων του hardware και να μην τα επιβαρύνει.

2.4.2. Επίπεδο Βιβλιοθηκών (Libraries)

Πάνω από τον πυρήνα Linux βρίσκεται το επίπεδο Βιβλιοθηκών, εκεί βρίσκεται ένα σύνολο βιβλιοθηκών, όπως το WebKit, του προγράμματος περιήγησης Web, η βιβλιοθήκη libc, η βάση δεδομένων SQLLite και άλλες. Οι βιβλιοθήκες αυτές είναι γραμμένες σε C/C++ και η πρόσβαση τους είναι μόνο δυνατή μέσω του Application Framework.

2.4.3. Επίπεδο Εκτέλεσης (Android Runtime)

Το Επίπεδο Εκτέλεσης το συναντάμε στο δεύτερο στρώμα λογισμικών από κάτω προς τα πάνω. Μέσα σε αυτό υπάρχει ένα πολύ σημαντικό στοιχείο το όποιο ονομάζεται Dalvik Virtual Machine, το οποίο είναι ένα είδος Java Virtual Machine σχεδιασμένο και προσαρμοσμένο στην πλατφόρμα Android. Το Dalvik VM κάνει χρήση βασικών λειτουργιών του πυρήνα Linux , όπως η διαχείριση μνήμης και του multi-threating. Η σημαντικότητα του όμως φαίνεται στο ότι δίνει την δυνατότητα στην κάθε εφαρμογή που χρησιμοποιεί την συσκευή να εκτελείται ξεχωριστά στο δικό της νήμα σαν να έχει το δικό της εικονικό μηχάνημα. Επιπλέον σε αυτό το επίπεδο υπάρχουν κάποιες

βιβλιοθήκες που μπορούν να τις χρησιμοποιήσουν οι προγραμματιστές για τον σχεδιασμό εφαρμογών μέσα από την Java.

2.4.4. Επίπεδο Πλαισίου Εφαρμογής (Application Framework)

Το επίπεδο Πλαισίου Εφαρμογής αποτελεί στο στρώμα αυτό το οποίο βοηθάει τους προγραμματιστές στην σχεδίαση και ανάπτυξη των εφαρμογών Android. Με αυτό τον τρόπο οι προγραμματιστές μπορούν να εκμεταλλευτούν όλες τις λειτουργίες της συσκευής όπως να έχουν πρόσβαση σε υπηρεσίες εντοπισμού θέσης, να εμφανίζουν ειδοποιήσεις και πολλά άλλα. Επίσης έχουν πρόσβαση στο ίδιο εύρος από APIs που έχουν οι βασικές εφαρμογές του Android. Αυτό οφείλετε στο γεγονός ότι η αρχιτεκτονική τους είναι με τέτοιο τρόπο σχεδιασμένη έτσι ώστε η καθεμία εφαρμογή να μπορεί να εκμεταλλευτεί τις δυνατότητες μιας άλλης. Οι σημαντικότερες υπηρεσίες που παρέχονται είναι:

- Activity Manager(Διαχειριστής εφαρμογών): Είναι η υπηρεσία αυτή η οποία διαχειρίζεται για πόσο μια δραστηριότητα θα παραμείνει ενεργή και δίνει την δυνατότητα να μπορεί κάποιος να μεταφέρεται από μια δραστηριότητα σε μια άλλη κρατώντας όμως αποθηκευμένη την σειρά με την οποία αυτές έχουν εκτελεστεί.
- Content Providers(Διαχειριστής περιεχομένων) : Είναι η υπηρεσία αύτη η οποία δίνει την δυνατότητα στις εφαρμογές να δημοσιοποιούν και να ανταλλάσσουν μεταξύ τους πληροφορίες.
- Resource Manager (Διαχειριστής Πόρων): Είναι η υπηρεσία αύτη η οποία δίνει πρόσβαση σε δεδομένα τα οποία δεν είναι σε μορφή κώδικα όπως string, εικόνες , οι κωδικοί των χρωμάτων και άλλα.
- Notification Manager (Διαχειριστής Ειδοποιήσεων): Είναι η υπηρεσία η οποία δίνει την δυνατότητα στις εφαρμογές να εμφανίσουν κάποιο μήνυμα στους χρήστες των εφαρμογών μέσα από notification ή alert.
- View System: είναι η υπηρεσία αυτή η οποία δημιουργεί το γραφικό περιβάλλον μιας εφαρμογής ,το οποίο περιλαμβάνει από grids και lists μέχρι buttons και spinners, όπως και πολλά αλλά για να ικανοποιηθούν οι ορέξεις του κάθε προγραμματιστή.

2.4.5. Επίπεδο Εφαρμογών (Applications)

Είναι το υψηλότερο επίπεδο στην στοίβα της αρχιτεκτονικής του λογισμικού μας. Σε αυτό το στρώμα βρίσκονται οι εφαρμογές οι οποίες εγκαθιστά και χρησιμοποιεί ο χρήστης ενός smartphone. Κάθε συσκευή έχει κάποιές προ εγκατεστημένες εφαρμογές για τις βασικές του λειτουργίες όπως του τηλεφώνου, της αποστολής μηνυμάτων ,του email client και άλλες. Επιπλέον μπορεί κάποιος να κατεβάσει και να εγκαταστήσει τις εφαρμογές της αρεσκείας του από το Google Play.

2.5. Γιατί Android

2.5.1. Λειτουργικότητα

Το Android είναι ένα λογισμικό το οποίο δίνει την ευχέρεια στον προγραμματιστεί να χρησιμοποιήσει προς όφελος του όλες αυτές τις δυνατότητες που του δίνει ένα απλό smartphone και να δημιουργήσει εφαρμογές οι οποίες βασίζονται σε στοιχεία τα οποία έχουν ήδη δημιουργηθεί. Για παράδειγμα άμα κάποιος θέλει να φτιάξει μια εφαρμογή για Gps Tracking το μόνο που χρειάζεται είναι να κατανοήσει τι λειτουργεία των βιβλιοθηκών που αφορά το Gps. Έτσι μπορούν να δημιουργηθούν πολλές εφαρμογές οι όποιες θα βασίζονται στην ίδια ιδέα αλλά θα έχουν διαφορετική υλοποίηση ανάλογα με τον τρόπο σκέψεις του εκάστοτε προγραμματιστή.

2.5.2. Ανάπτυξη Εφαρμογών

Η Google από μόνη της δημιούργησε με τέτοιο τρόπο την αρχιτεκτονική του λογισμικού έτσι ώστε να ενθαρρύνει όλο και περισσότερους προγραμματιστές να ασχοληθούν με την σχεδίαση και ανάπτυξη εφαρμογών για την πλατφόρμα της. Αυτό το πέτυχε με την δημοσιοποίηση του κώδικα του λογισμικού αλλά και με την ύπαρξή μια μεγάλής ποικιλίας από βιβλιοθήκες τις οποίες μπορεί να τις χρησιμοποιήσει ο προγραμματιστής για να δημιουργήσει την εφαρμογή που επιθυμεί με ευκολία εκμεταλλευόμενος λειτουργίες είτε τις οποίες έχει αναπτύξει η ίδια η εταιρία ή κάποιος άλλος προγραμματιστής.

2.5.3. Κόστος Ανάπτυξης

Για την δημιουργία κάποιας εφαρμογής στα πρωταρχικά της στάδια δεν χρειάζεται δαπανήσει πολλά χρήματα. Το μόνο που χρειάζεται κάποιος προγραμματιστής είναι να έχει στην διάθεση του κάποια συσκευή android, η οποία και πάλι μπορεί να την βρει σε πολύ προσιτές τιμές ανάλογα βέβαια και τις ανάγκες που θέλει να καλύψει με την αγορά της, και ένα από τα δωρεάν προγράμματα λογισμικού για σχεδίαση και ανάπτυξη εφαρμογών android όπως το Android Studio.

2.5.4. Γλώσσα Προγραμματισμού

Οι εφαρμογές Android χρησιμοποιούν την γλώσσα προγραμματισμού Java. Μια ευρέως γνωστή γλώσσα προγραμματισμού την οποία χρησιμοποιούν οι περισσότεροι προγραμματιστές στην σήμερον ήμερα και τους δίνει την ευκαιρία να αναπτύξουν εύκολα μια εφαρμογή χωρίς να χάσουν χρόνο για την εκμάθηση κάποιας νέας γλώσσας προγραμματισμού.

Κεφάλαιο 3

Τεχνολογίες που χρησιμοποιήθηκαν

Περιεχόμενα Κεφαλαίου

3.1. Web Services

3.1.1. Τι είναι τα Web Services

Είναι γεγονός ότι, πολλοί ορισμοί έχουν διατυπωθεί, προσπαθώντας να αποδώσουν του τι είναι τα Web Services. Ένας ορισμός ο οποίος είναι αρκετά ικανοποιητικός και δίνεται από το W3C είναι ο εξής:

" Ένα Web Service, είναι ένα σύστημα λογισμικού σχεδιασμένο να υποστηρίζει τη διαλειτουργική, μηχανή προς μηχανή, διάδραση μέσω ενός δικτύου. Έχει μια διεπαφή με περιγραφή που είναι επεξεργάσιμη από μηχανές (συγκεκριμένα, βάση της περιγραφής WSDL- Web Service Definition Language). Τα άλλα συστήματα διαδρούν με τα Web Service με τον τρόπο με τον οποίο καθορίζεται από την περιγραφή του Web Service, χρησιμοποιώντας μηνύματα SOAP, τα οποία κανονικά μεταδίδονται με την χρήση HTTP- HypeText Transfer Protocol(με μια κωδικοποίηση σε XML σε συνδυασμό με άλλα Web πρότυπα."

Επιπλέον η IBM έχει διατυπώσει και αύτη έναν ορισμό ο οποίος είναι πιο ολοκληρωμένος και κατανοητός:

" Τα Web Services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα Web Service είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μία λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από web services οι οποίες αλληλοεπιδρούν μεταξύ τους καθορίζει μια εφαρμογή web services σύμφωνα με την προσανατολισμένη στις Υπηρεσίες Αρχιτεκτονική (Service-Oriented Architecture – SOA).»

Από τους δύο παραπάνω ορισμούς, μπορούμε να συμπεράνουμε ότι ένα Web Service περιέχει μια αρχιτεκτονική κατανεμημένων συστημάτων από πολλά διαφορετικά υπολογιστικά συστήματα. Τα Web Service μπορεί να είναι γραμμένο σε διαφορετική γλώσσα προγραμματισμού ,σε σχέση με το σύστημα που το καλεί, και επιτελεί ξεχωριστή λειτουργία. Τα παραπάνω συστήματα, επικοινωνούν μεταξύ τους μέσω του διαδικτύου ώστε να δημιουργήσουν ένα νέο ευρύτερο σύστημα. Επομένως είναι πολύ χρήσιμα για τους προγραμματιστές γιατί τους βοηθάνε

να υλοποιήσουν κατανεμημένες εφαρμογές, κάνοντας χρήση μία μεγάλης ποικιλίας από εργαλεία, έτσι ώστε να δημιουργήσουν τα δικά τους λογισμικά.

Γενικά υπάρχουν 2 ειδών Web Services:

I. Αυτές που η μόνη τους λειτουργία είναι η ρητή αναμονή για να έρθει μια αίτηση από κάποιον client, να την επεξεργαστούν και να προετοιμάσουν την κατάλληλη απάντηση.
II. Αυτές που είναι πιο πολύπλοκες και προσπαθούν να συντονίσουν διάφορες υπηρεσίες μεταξύ τους.

3.1.2. Αρχιτεκτονική REST

Ο όρος REST προέρχεται από τις λέξεις Representational State Transfer και αφορά μια αρχιτεκτονική υλοποίηση υπηρεσιών ανάμεσα σε κατανεμημένα συστήματα υπερμέσων. Η παραπάνω αρχιτεκτονική υλοποίηση, χρησιμοποιεί HTTP Protocol για την επικοινωνία μεταξύ των μηχανημάτων. Σκοπός αυτής της υλοποίησης, αποτελεί η ένωση διαφόρων μηχανημάτων, τα οποία επιτελούν διαφορετικές λειτουργίες. Μέσω της αρχιτεκτονικής Rest, επιδιώκεται τα μηχανήματα που αναφέρθηκαν παραπάνω να πάρουν τους πόρους που χρειάζονται για να επιτελέσουν το έργο τους (έχοντας πάντοτε το ρόλο του middleware και χρησιμοποιώντας HTTP μεθόδους για όλες τις αιτήσεις και απαντήσεις που πραγματοπούνται) . Τα Web Services που υλοποιούνται πάνω σε αύτη την αρχιτεκτονική, ονομάζονται RESTful WebServices ή RESTful API. Το πρότυπο εμφανίστηκε πρώτη φόρα το 2000 από τον Roy Fieldind, όταν στην διδακτορική του

Rest WebService

διατριβή προσπάθησε να περιγράψει την αρχιτεκτονική δομή του διαδικτύου.

Η αρχιτεκτονική ενός REST είναι σχετικά απλή και εύκολη να σχεδιαστεί. Αυτό που κάνει ο REST Server, είναι απλά να παρέχει πρόσβαση σε πόρους, όπως αυτούς μιας βάσης δεδομένων, έτσι ο client να μπορεί να έχει πρόσβαση σε αυτούς τους πόρους και να τους επεξεργαστεί κατάλληλα.

Για να μπορέσει ο client να καλέσει τον RESTful API, υπάρχουν κάποιοι κανόνες οι οποίοι ορίζονται στο documentation του καθενός, με σημαντικότερο να αποτελεί τον τρόπο επικοινωνίας ανάμεσα στα δύο συστήματα έχοντας ως σημείο αναφοράς ένα URI (Uniform Resource Identifier). Ένα URI χρησιμοποιείται για να προσδιορίσει ποια λειτουργία θέλει να καλέσει ο client και να ορίσει κάποιες συγκεκριμένες παραμέτρους σε περίπτωση που αυτές

χρειάζονται. Έτσι ώστε να πάρει τα κατάλληλα αποτέλεσμα από το Web Service. Η γνωστότερη μορφή URI στο διαδίκτυο είναι τα URL.

Όσον αφορά το reply του RESTful API είναι και αυτό ορισμένο στο documentation, δίνοντας τη δυνατότητα στον client να διαλέξει την μορφή που τον εξυπηρετεί. Συνήθως η αναπαράσταση είναι σε κάποια μορφή κειμένου όπως οι JSON και XML, με τους περισσότερους να χρησιμοποιούν JSON.

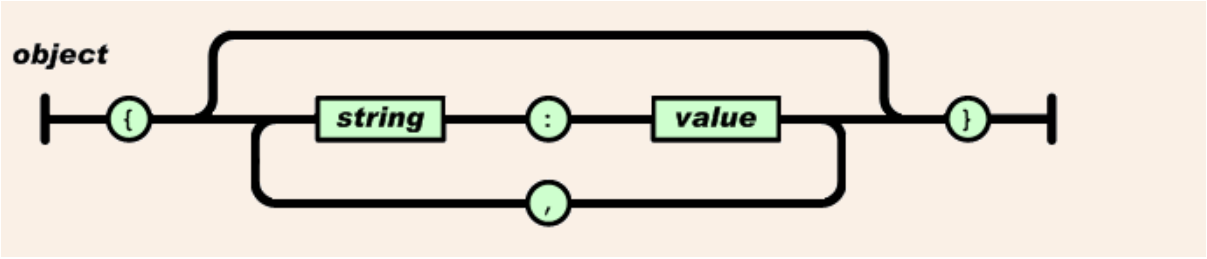
3.1.3. JSON

Όπως είδαμε προηγουμένως μια από τις μορφές με τις οποίες μπορεί να απαντήσει ένα RESTful API είναι η κλάση JSONObject.

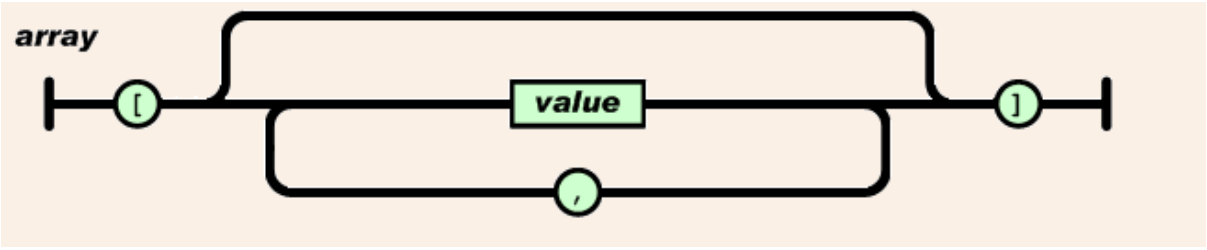
Τα ακρωνύμια της λέξης JSON προέρχονται από τις λέξεις JavaScript Object Notation.Το JSON είναι μια ελαφριά μορφή ανταλλαγής δεδομένων. Ο λόγος για τον οποίο χρησιμοποιείται σε μεγάλο βαθμό, οφείλεται στο ότι είναι ευκολά κατανοητή από τους ανθρώπους και από τα λογισμικά, αλλά και επειδή είναι εύκολη και στο γράψιμο και στην παραγωγή. Πρόκειται για μια μορφή κειμένου, η οποία είναι τελείως απλή χρησιμοποιεί όμως συμβάσεις για να μπορεί να χρησιμοποιείται από γλώσσες προγραμματισμού με κυριότερες τις C,C++, Python ,Java. Όλα αυτά τα χαρακτηριστικά την καθιστούν ιδανική για την ανταλλαγή δεδομένων.

Υπάρχουν δυο βασικές δομές από τις οποίες μπορεί να χτιστεί μια μεταβλητή τύπου JSON:

- Μια συλλογή από δεδομένα τα οποία αποθηκεύονται σε ζευγάρια, με την ονομασία η οποία τους δίνεται να είναι JSONObject. Η μορφή τους μοιάζει σε αυτή της Εικόνας 3.1
- Μία διατεταγμένη λίστα τιμών, οι οποίες αποθηκεύονται μέσα σε έναν πίνακα για το λόγο αυτό έχουν την ονομασία JSONArray. Η μορφή τους μοιάζει σε αυτή της Εικόνας 2.



Εικόνα 3.1: JSONObject



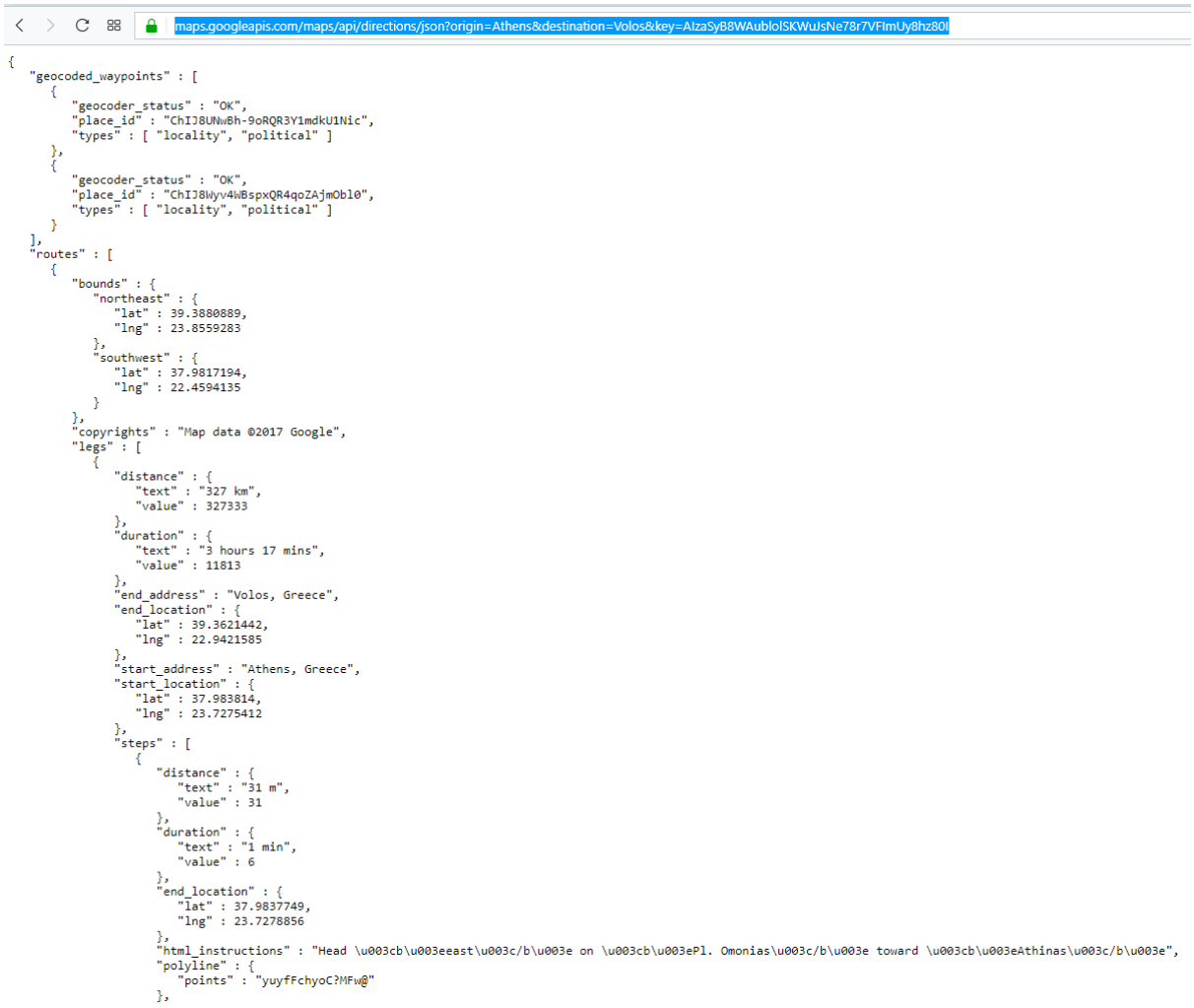
Εικόνα 3.1:JSONArray

3.2.Google API

Το Google API είναι από τα πιο χρήσιμα API που υπάρχουν αύτη την στιγμή και μπορεί να χρησιμοποιήσει ένας προγραμματιστής. . Ο λόγος για τον οποίο καθίσταται τόσο χρήσιμο, οφείλεται στο ότι είναι σχεδιασμένο από την Google για να μπορείς να έχεις πρόσβαση σε βασικές υπηρεσίες που έχει αναπτύξει και να έχεις τι δυνατότητα να τις χρησιμοποιήσεις στην εφαρμογή σου όπως το Google search, Google Maps και το Google translate. Επομένως οι εφαρμογές είτε αφορούν smartphone είτε web app μπορούν να τις χρησιμοποιήσουν και να εκμεταλλευτούν πολλά από τα πλεονεκτήματα που δίνουν οι συγκεκριμένες ή και να τα επεκτείνουν ακόμα. Για παράδειγμα άμα χρειάζεται να φτιαχτεί μια εφαρμογή η οποία θέλει να έχει ένα login form μπορείς να έχεις επιλογή ο χρήστης να συνδέεται με τα στοιχεία που έχει στο google++.

Στην παρούσα διπλωματική εργασία τα APIs χρησιμοποιήθηκε το Google Maps API το οποίο βοήθησε να πάρουμε καίριες πληροφορίες που χρειαζόταν για την περάτωση της εφαρμογής μας. Η κυριότερη λειτουργία στην οποία μας βοήθησε ήταν να μας δίνει τις οδηγίες για να φτάσουμε από ένα σημείο σε ένα άλλο. Επίσης με την βοήθεια του μπορέσαμε να πάρουμε και πληροφορίες όπως η απόσταση μεταξύ των δύο σημείων, το χρόνο που χρειάζεσαι για να πας από ένα σημείο σε ένα άλλο, αλλά και την διεύθυνση την στην οποία βρίσκεται ένα σημείο.

Αυτό το API είναι πολύ χρήσιμο και εύκολο στο να το χρησιμοποιήσεις. Για την χρήση του το μόνο που χρειάζεται κανείς είναι να καλέσει την παρακάτω τοποθεσία «https://maps.googleapis.com/maps/api/directions/outputFormat?parameters». Το output parameter είναι η μορφή στην οποία θέλεις να είναι η απάντηση σου οι επιλογές που έχει είναι είτε σε μορφή JSON, η οποία είναι και η προτεινόμενη, είτε σε xml. Όσον αφορά τώρα το parameter είναι της μορφής «origin=Athens&destination=Volos&key=YOUR_API_KEY» όπου origin είναι η αρχική θέση της διαδρομής σου και destination είναι η τελική θέση. Επιπλέον για να κάνεις request στο Google API χρειάζεται να έχεις account στην Google και να πάρεις ένα κλειδί το οποίο θα είναι μοναδικό για το project σου αυτή είναι του key μέσα στο parameter. Την κλήση αύτη μπορείς να την κάνεις είτε μέσα από κάποια εφαρμογή είτε αύτη είναι για smartphone ή από κάποια web app ή από κάποιον server άλλα και απλά να την πληκτρολογήσεις στον browser. Άμα κληθεί από browser η απάντηση θα φανεί σε μορφή HTML και θα είναι όπως στην Εικόνα 3.3, σε διαφορετική περίπτωση υπάρχει η επιλογή από τον προγραμματιστή της μορφής όπως αναφέρθηκε και προηγουμένως και κατάλληλος χειρισμός.



Εικόνα 3.2 Κλήση Google Maps API και Απάντηση σε browser

Κεφάλαιο 4

Εργαλεία που χρησιμοποιήθηκαν

Περιεχόμενα Κεφαλαίου

4.1.Android Studio

Είναι το επίσημο λογισμικό το οποίο συνιστά η Google για την σχεδίαση και ανάπτυξη εφαρμογών Android, το οποίο βασίστηκε στο JetBrains' IntelliJ IDEA software. Η παρουσίασή του έγινε στις 16 Μάιου 2013 στο συνέδριο της Google και είχε ως σκοπό να αντικαταστήσει το Eclipse Android Development Tools (ADT) ως κύριο IDE της Google για την ανάπτυξη Android εφαρμογών.

4.1.1. Δομή project

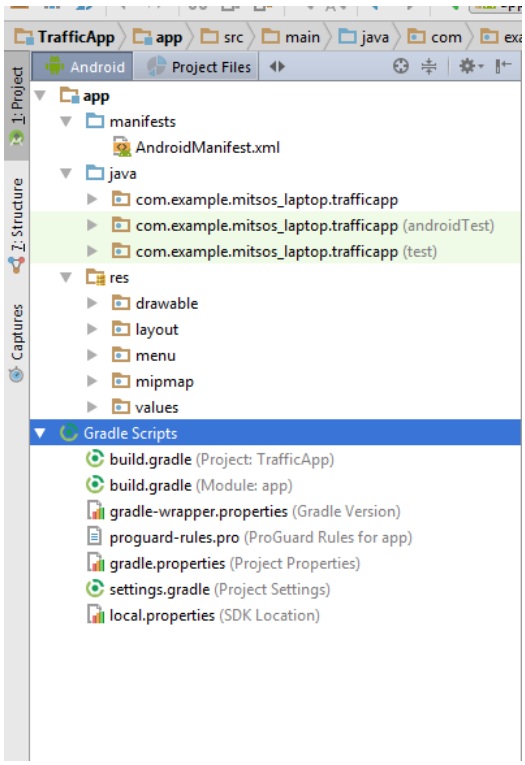
Κάθε project περιέχει ένα ή περισσότερα module μέσα στα οποία βρίσκεται ο κώδικας που δημιουργεί την εφαρμογή. Υπάρχουν 3 ειδών modules:

- Android app Modules: είναι το module αυτό το οποίο παρέχει έναν container για τον κώδικα της εφαρμογής ,τα resource files και τις ρυθμίσεις της εφαρμογής που θα βοηθήσουν την εφαρμογή να δημιουργηθεί και να εγκατασταθεί στο smartphone.
- Library Module: είναι το module αυτό το οποίο παρέχει έναν container για τον επαναχρησιμοποιούμενο κώδικα, ο οποίος μπορεί να χρησιμοποιηθεί είτε ως dependency σε άλλα module της εφαρμογής ή σαν import σε άλλα project.
- Google Cloud module : είναι το module αυτό το οποίο παρέχει έναν container για Google Cloud backend κώδικα της εφαρμογής.

Κατά κανόνα η οργάνωση και ο τρόπος που εμφανίζονται τα αρχεία ενός project μοιάζει με αυτή της εικόνας 4.1.

Κάθε project για να μπορέσει να δημιουργηθεί σωστά περιέχει τους εξής φακέλους:

- manifest: περιέχει το AndroidManifest.xml στο οποίο ορίζονται όλα τα δικαιώματα της εφαρμογής και οι οθόνες που περιέχει η εφαρμογή.
- java : περιέχει τον κώδικα Java οποίος τρέχει στην εφαρμογή.
- res: περιέχει όλα αυτά τα non-code resources , όπως τα XML layout, UI,strings και άλλα.



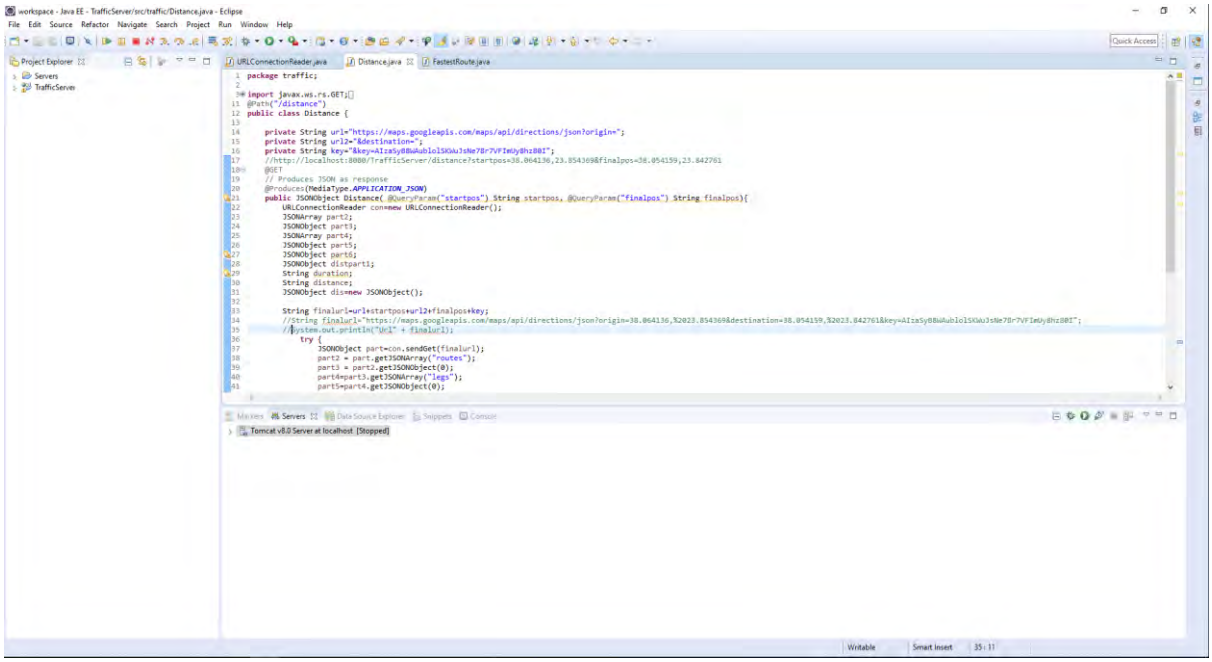
Εικόνα 4.1: Οργάνωση αρχείων

4.2.Eclipse

Το Eclipse είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης έργων λογισμικού, το οποίο χρησιμοποιείται κυρίως για την δημιουργία εφαρμογών σε Java. Ωστόσο με την προσθήκη του κατάλληλου compiler μπορούν να δημιουργηθούν προγράμματα από άλλες γλώσσες όπως C++, Python, PHP και άλλες. Αναπτύχθηκε από την IBM και έκανε την εμφάνιση του πρώτη φορά το 2001. Πλέον την ανάπτυξη και βελτιστοποίηση του έχει αναλάβει ο μη κερδοσκοπικός οργανισμός Eclipse Foundation.

Το Eclipse ως λογισμικό είναι ελεύθερο και ο κώδικας του είναι ανοιχτός προς ανάπτυξη και βελτιστοποίηση. Έχει εκδοθεί κάτω από τους όρους του Eclipse Public License, αλλά δεν είναι συμβατός με το αυτόν του GNU(General Public License).

Εικόνα 4.2: Eclipse Logo



Εικόνα 4.3: Eclipse User Interface

4.3.XAMPP

Το XAMPP είναι ένα ελεύθερο και ανοιχτό λογισμικό το οποίο χρησιμοποιείται ανεξαρτήτου πλατφόρμας και περιέχει ένα πακέτο λύσεων για την δημιουργία web servers. Δημιουργήθηκε από την Apache Friends και αποτελείται κυρίως από έναν Apache HTTP Server, τη MariaDB database και μεταφραστές για script γραμμένα σε PHP και Perl. Είναι μια απλή και ελαφριά έκδοση του Apache που βοηθάει τους προγραμματιστές να στήσουν έναν τοπικό web server για να μπορέσουν να αναπτύξουν τις εφαρμογές τους και να τις δοκιμάσουν. Επιπλέον είναι μια πλατφόρμα η οποία είναι συμβατή με όλα τα

λογισμικά δηλαδή, Linux, Mac και Windows. Τέλος καθοριστικό παράγοντα στην ευρεία χρήση του λογισμικού αυτού αποτελεί η ύπαρξη συστατικών ίδιων με εκείνων ενός online web server, με αποτέλεσμα η μετάβασή του σε αυτόν να γίνετε πολύ εύκολα.

Εικόνα 4.4: Λογότυπο XAMPP και Βασικά Components

Τα ακρωνύμιά του προέρχονται από τα παρακάτω συστατικά:

- X: προέχεται από την έννοια “cross-platform”, δηλαδή ένα λογισμικό ανεξάρτητου πλατφόρμας.
- A: προέχεται από τον Apache HTTP Server
- M: προέχεται από την έκδοση της βάσης δεδομένων που χρησιμοποιεί Maria DB
- P: προέχεται από τα PHP script που μπορούν να μεταφραστούν.
- P: προέχεται από τα Perl script που μπορούν να μεταφραστούν.

4.4. PHPMyAdmin

Το PHPMyAdmin είναι ένα ελεύθερο εργαλείο λογισμικού γραμμένο σε PHP με σκοπό τη διαχείριση της εκάστοτε βάσης δεδομένων που υπάρχει σε έναν server. Η διεργασία αυτή επιτυγχάνεται μέσω ενός ευρέος φάσματος λειτουργειών που περιέχονται στην MySql και την MariaDB. Οι λειτουργίες αυτές, όπως η δημιουργία βάσεων, table, columns και πολλές άλλες, μπορούν να πραγματοποιηθούν μέσω του γραφικού περιβάλλοντος, ωστόσο δίνεται επιπλέον και η δυνατότητα εκτέλεσης εντολών SQL.

Εικόνα 4.5: Λογότυπο PHPMyAdmin

Είναι ένα από τα πιο διαδεδομένα λογισμικά για διαχείριση βάσεων δεδομένών και έχει μεταφραστεί σε 72 γλώσσες ένω υποστηρίζει τόσο LTR και RTL γλώσσες.

Το project PHPMyAdmin είναι μέλος του Software Freedom Conservancy. Το SFC είναι μια μη κερδοσκοπική οργάνωση η οποία βοηθάει στην προώθηση, την ανάπτυξη, την βελτίωση και την προστασία των Free, Libre, and Open Source Software (FLOSS) projects.

4.5.Σύνοψη

Για τη σχεδίαση και υλοποίηση της εφαρμογής χρησιμοποιήθηκε το λογισμικό πρόγραμμα Android Studio 2.3.3, με βασική γλώσσα προγραμματισμού να αποτελεί αύτη της Java.

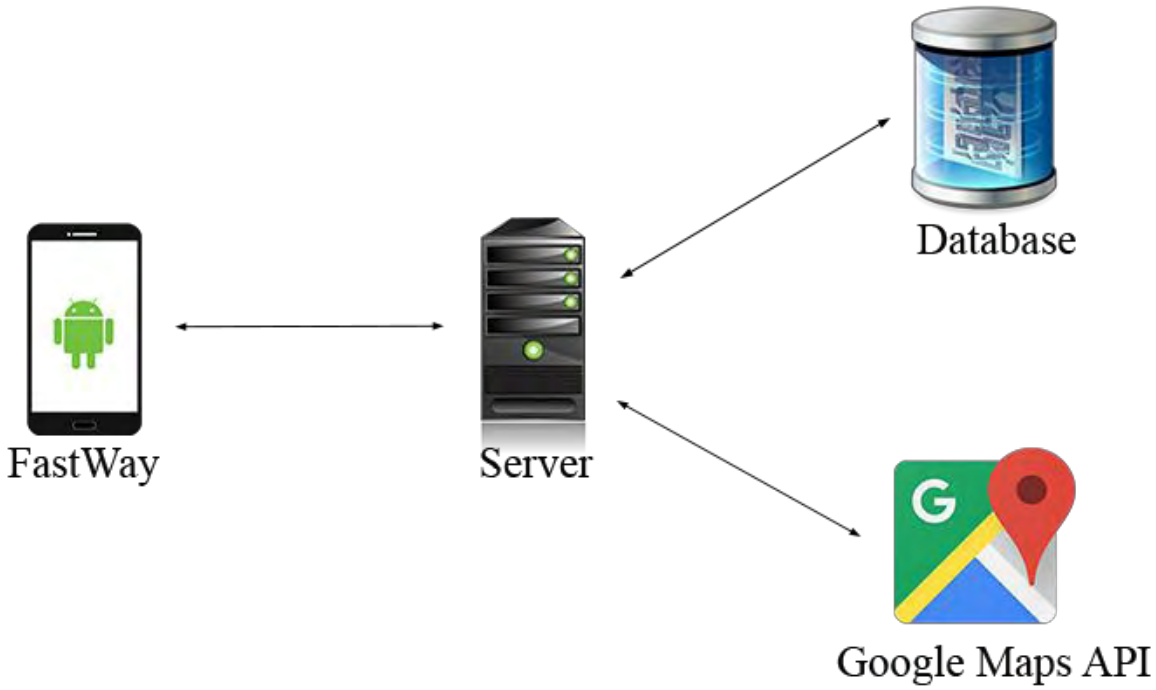
Για την ανάπτυξη του Web Service χρησιμοποιήθηκε το Eclipse Java EE IDE for Web Developers Neon 3 καθώς και το λογισμικό XAMPP. Στην αρχή η ανάπτυξη έγινε τοπικά προκυμμένου να ελεγχθεί η σωστή λειτουργία του server και η σύνδεση του με την εφαρμογή. Για την εγκυρότερη συλλογή δεδομένων κατέστη αναγκαία η μετατροπή του τοπικού server σε online. Η διαδικασία αυτή επετεύχθη με την δημιουργία Static IP και Port Forwarding.

Κεφάλαιο 5

Αρχιτεκτονική Συστήματος

Περιεχόμενα Κεφαλαίου

5.1.Σχέδιο Υλοποίησης



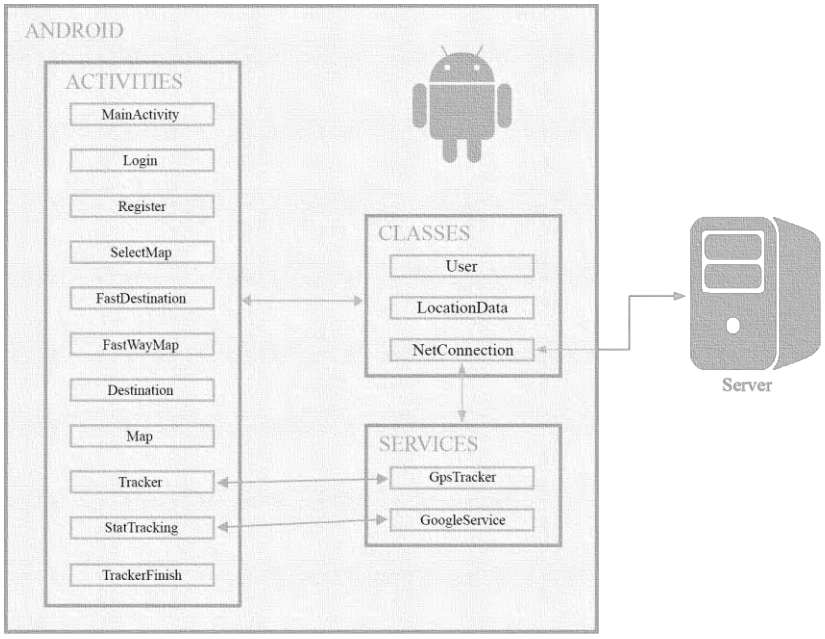
Εικόνα 5.1: Δομή Αρχιτεκτονικής

Για την σωστή λειτουργία της παρούσας διπλωματικής εργασίας, χρειάστηκαν να συνδυαστούν ορισμένες επιμέρους υπομονάδες μεταξύ τους. Στην Εικόνα 5.1, παρουσιάζεται η αρχιτεκτονική του συστήματος. Τα δύο βασικά τμήματα της συγκεκριμένης αρχιτεκτονικής είναι η υλοποίηση της εφαρμογής και του εξυπηρετητή (server).

- Η ανάπτυξη της παρούσας εφαρμογής, πραγματοποιείται στο λειτουργικό σύστημα Android. Πρόκειται για μια native εφαρμογή η οποία είναι σχεδιασμένη στο Android Studio, όπως έχει αναφερθεί και παραπάνω. Οι κύριες λειτουργίες της εφαρμογής είναι η συλλογή δεδομένων που έχουν σχέση με την τρέχουσα θέση του χρήστη και η παρουσίαση της συντομότερης διαδρομής στην οθόνη του smartphone του χρήστη. Οι λειτουργίες αυτές επιτυγχάνονται μέσω της ανταλλαγής δεδομένων με τον εξυπηρετητή (Server).
- Ο server, σχεδιάστηκε με τη βοήθεια του ελεύθερου λογισμικού XAMPP. Η κυριότερη λειτουργία του server, είναι η λήψη δεδομένων που αποστέλλονται από την εφαρμογή και η αποθήκευση τους στην βάση δεδομένων. Τα δεδομένα αυτά αποτελούν καίριες πληροφορίες για την επίτευξη του τελικού σκοπού, ο οποίος αφορά την εύρεση της συντομότερης διαδρομής που έχει ζητηθεί από τον χρήστη.
- Το Google Maps Api χρησιμοποιείται τόσο από τον server όσο και από την εφαρμογή για την σωστή και εύρυθμη λειτουργία τους.

5.2.Ανάλυση Υπομονάδων

5.2.1. Εφαρμογή Android



Εικόνα 5.2: Δομή Android εφαρμογών

Η εφαρμογή αυτή, απευθύνεται σε όλους τους ανθρώπους οι οποίοι έχουν στην κατοχή τους Android smartphones και επιθυμούν να βρουν την ταχύτερη διαδρομή για τον προορισμό τους.Ρο Για την χρήση των λειτουργιών που προσφέρει η εφαρμογή, απαραίτητη προϋπόθεση, αποτελεί η εγγραφή του χρήστη στην εφαρμογή. Να σημειωθεί ότι, ο χρήστης μπορεί είτε να συμβάλει στην επέκταση των ορίων της είτε απλά να την χρησιμοποιήσει για δικό του όφελος.

Ο χρήστης μπορεί να συμβάλει στην εφαρμογή δίνοντας πληροφορίες σχετικά με τις τοποθεσίες του κατά την διάρκεια μιας διαδρομής που πραγματοποιεί. Η λειτουργία αύτη επιτυγχάνεται μέσω των Activities Tracker, StatTracking και TrackerFinish. Αρχικά ο χρήστης θα χρειαστεί να δηλώσει τον προορισμό του έτσι ώστε η εφαρμογή να αποστείλει στον server την διαδρομή που θα καταγραφεί. Κατά την διάρκεια της διαδρομής σε τακτά χρονικά διαστήματα (2 λεπτά) καταγράφονται οι εκάστοτε τοποθεσίες του χρήστη, οι οποίες αποστέλλονται στο server, έτσι ώστε να αντλούνται πληροφορίες για την χιλιομετρική απόσταση που διένυσε μέσα στο χρονικό διάστημα αυτό. Όταν ο χρήστης φτάσει στον προορισμό του η λειτουργία αύτη ολοκληρώνεται και εμφανίζονται τα στατιστικά στοιχεία της διαδρομής του (TrackingFinish).

Σε αντίθετη περίπτωση ο χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή μόνο για δικό του όφελος μέσω των Activities FastDestinantion και FastWayMap, που αφορούν την εμφάνιση της συντομότερης διαδρομής στον χάρτη. Η οθόνη FastDestinantion επιτρέπει στο χρήστη να επιλέξει την διαδρομή που εκείνος επιθυμεί. Οι διαδρομές αυτές είναι προκαθορισμένες και παρέχονται από το server. Η εμφάνιση της διαδρομής στην οθόνη FastWayMap προέρχεται από δεδομένα Ποιο συγκεκριμένα, η οθόνη FastDestinantion, επιτρέπει στο χρήστη να επιλέξει την διαδρομή που εκείνος επιθυμεί, με τις διαδρομές αυτές να είναι προκαθορισμένες, προερχόμενες από το server. Η εμφάνιση της διαδρομής στην οθόνη FastWayMap, πραγματοποιείται με βάση τα

δεδομένα που έχουν ληφθεί από το server και βοηθούν στην απεικόνιση της βέλτιστης διαδρομής στους χάρτες της Google.

Τέλος να σημειωθεί ότι, σε περίπτωση που δημιουργηθεί αδυναμία εύρεσης της βέλτιστης διαδρομής λόγω κάποιου σφάλματος, δίνεται η δυνατότητα στο χρήστη να επανεισάγει την επιθυμητή διαδρομή, ώστε να εμφανιστεί η προτεινόμενη διαδρομή της Google (Destination, Map).

5.2.2. Εξυπηρετητής (Server)

Ο server, αποτελεί το ενδιάμεσο στάδιο μεταξύ της εφαρμογής και της βάσης δεδομένων. Σε κάποιες από τις λειτουργίες της χρειάζεται να καλέσει και το Google Maps API για να λάβει ορισμένες πληροφορίες που χρειάζονται.

Μια από τις βασικότερες λειτουργίες η οποία επιτελείται στον server, είναι η κλήση της υπηρεσίας CarStats. Μέσα από την υπηρεσία αυτή μπορούν να αποθηκευτούν οι πληροφορίες που σχετίζονται με την κίνηση στους δρόμους. Αυτό επιτυγχάνεται με την αποθήκευση της απόστασης μεταξύ δύο σημείων κατά την διάρκεια μιας διαδρομής.

Σημαντικό ρόλο διαδραματίζει και η λειτουργία FasterRoute, στην οποία βρίσκεται ο αλγόριθμος μέσα από τον οποίο βρίσκεται η γρηγορότερη αλλαγή. Ο αλγόριθμος αυτός, θα εκμεταλλευτεί τις πληροφορίες που έχουμε συλλέξει και σχετίζονται με την κίνηση στους δρόμους.

Επιπλέον, στον server υπάρχουν και κάποιες λειτουργίες οι οποίες δεν είναι εξίσου σημαντικές όσο οι προαναφερθείσες, ωστόσο βοηθάνε στην εύρυθμη λειτουργία της εφαρμογής και του server. Οι λειτουργίες αυτές είναι οι εξής:

- Register-Login: Οι λειτουργίες αυτές που σχετίζονται με την εγγραφή και την είσοδο των χρηστών στην υπηρεσία.
- Distance: Η λειτουργία η οποία επιστρέφει την διαδρομή δύο σημείων με την βοήθεια του Google Maps Api
- AllDestinations: η λειτουργία η οποία επιστρέφει όλες τις διαδρομές που έχουν αποθηκευτεί στην βάση δεδομένων.

5.2.3. Βάση Δεδομένων

Εικόνα 5.4: Βάση Δεδομένων

Η βάση δεδομένων, αποτελεί ένα πολύ σημαντικό κομμάτι της συγκεκριμένης διπλωματικής εργασίας. Μέσω αυτής υπάρχει καθίσταται δυνατό στο να επικοινωνεί μόνο ο server, ο οποίος αποθηκεύει και τα δεδομένα σε αύτη.

Για την παρούσα διπλωματική εργασία έχουν δημιουργηθεί 3 tables τα οποία είναι τα εξής:

- Account: πρόκειται για το table στο οποίο αποθηκεύονται τα δεδομένα των χρηστών της εφαρμογής . Έχει 2 columns τα οποία είναι:
  - Username : σε αύτη την στήλη αποθηκεύονται όλα τα ονόματα των χρηστών της εφαρμογής
  - Password: σε αύτη την στήλη αποθηκεύεται ο κωδικός εισόδου του εκάστοτε χρήστη για την πρόσβαση του στην υπηρεσία.
- CarStats: πρόκειται για το table στο οποίο αποθηκεύονται οι πληροφορίες για την τρέχουσα θέση του χρήστη. Σε αυτό εμπεριέχονται τα εξής 8 columns:
  - Startpos: αφορά την θέση από την οποία ξεκίνησε η μέτρηση.
  - Finalpos: αφορά την θέση στην οποία βρίσκεται την τρέχουσα χρονική στιγμή.
  - Idealtime: αφορά τον ιδανικό χρόνο ιδανικός χρόνος ο οποίος δίνεται από το Google Maps για να καλυφθεί η συγκεκριμένη απόσταση.
  - Ourtime: ο χρόνος που χρειάστηκε ο χρήστης μας για να καλύψει την συγκεκριμένη απόσταση.
  - Daytime: η ώρα της ημέρας στην οποία πάρθηκαν τα συγκεκριμένα δεδομένα
  - Day: η μέρα η οποία πήραμε τα συγκεκριμένα δεδομένα
  - Distance: η απόσταση η οποία κάλυψε ο χρήστης μας
  - Id: αφορά το σε ποια διαδρομή ανήκει αύτη η απόσταση
- Destinations: σε αυτό το table αποθηκεύονται όλες οι διαδρομές για τις οποίες έχουν εγγράψει στοιχεία οι χρήστες στην βάση μας. Έχουμε 3 columns οι οποίες είναι:
  - Startpos: η αρχική θέση του προορισμού
  - Finalpos: ο προορισμός
  - Id: το αναγνωριστικό για το ποια διαδρομή είναι αύτη που θα αποθηκευτεί.

Κεφάλαιο 6

Υλοποίηση Συστήματος

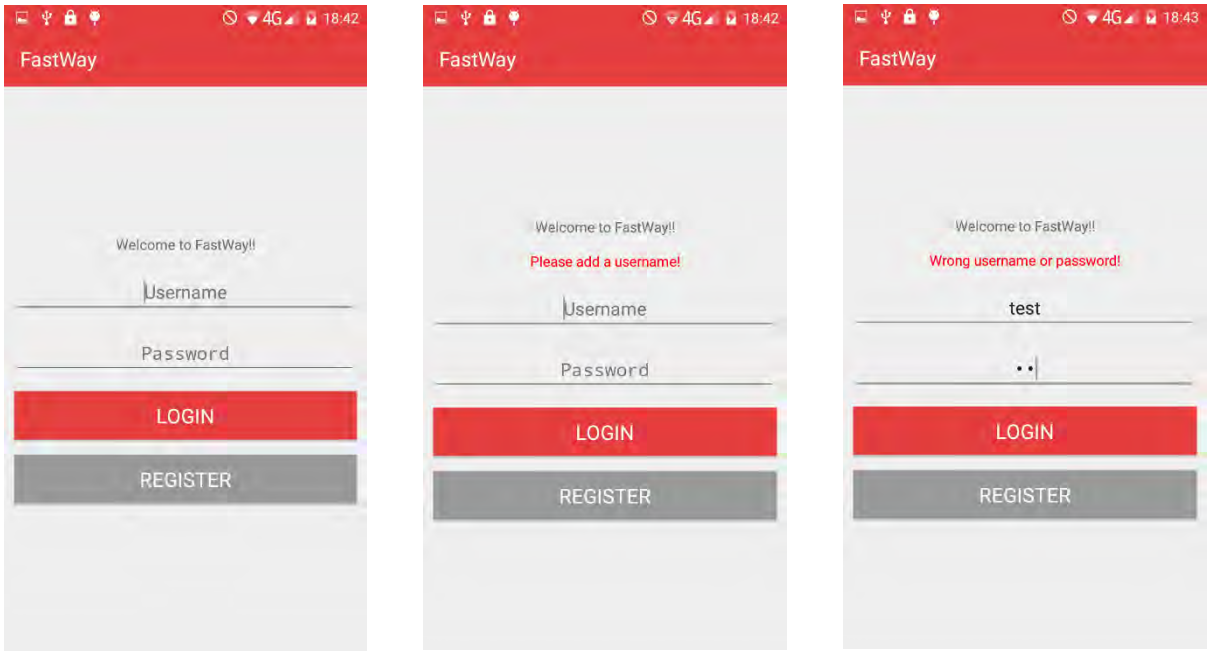
Περιεχόμενα Κεφαλαίου

6.1.FastWay

6.1.1. Παρουσίαση Εφαρμογής

Εικόνα 6.1 FastWay Icon

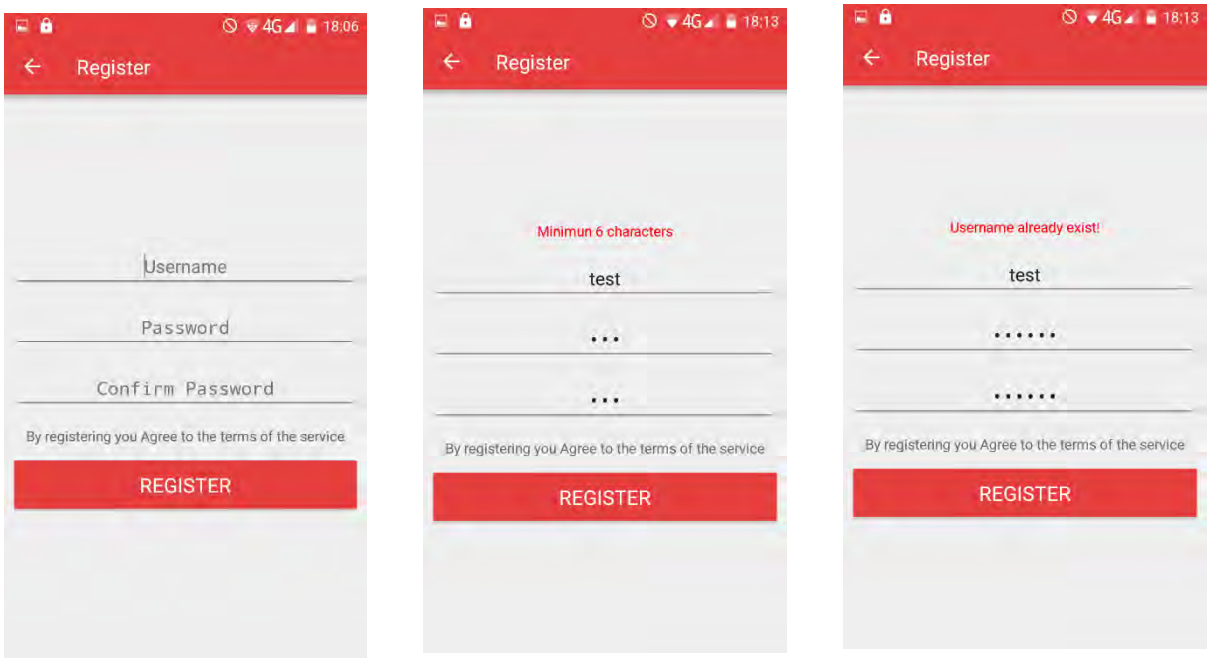
Η εφαρμογή που αναπτύχθηκε, σχεδιάστηκε με σκοπό να βοηθήσει τους χρήστες να φτάνουν συντομότερα στον προορισμό τους. Απαραίτητη προϋπόθεση για να επιτευχθεί αυτός ο σκοπός, αποτέλεσε η ενσωμάτωση κατάλληλων λειτουργιών οι οποίες ως στόχο έχουν την περισυλλογή δεδομένων για την κίνηση των δρόμων. Μέσω των δεδομένων που θα συλλέγονταν, θα προκύπταν σημαντικά συμπεράσματα για το ποιος δρόμος είναι καταλληλότερος να προταθεί στον χρήστη, έτσι ώστε να τον βοηθήσει να εξοικονομήσει χρόνο και να φτάσει γρηγορότερα στον προορισμό του. Τέλος να σημειωθεί ότι η εφαρμογη που αναπτύχθηκε θα έχει, το όνομα FastWay, το οποίο είναι βασισμένο στο βασικό πρόβλημα που προσπαθεί να λύσει η εφαρμογή, το οποίο είναι η εύρεση της της γρηγορότερης διαδρομής.



Εικόνα 6.2: Login interface και μηνύματα σφάλματος

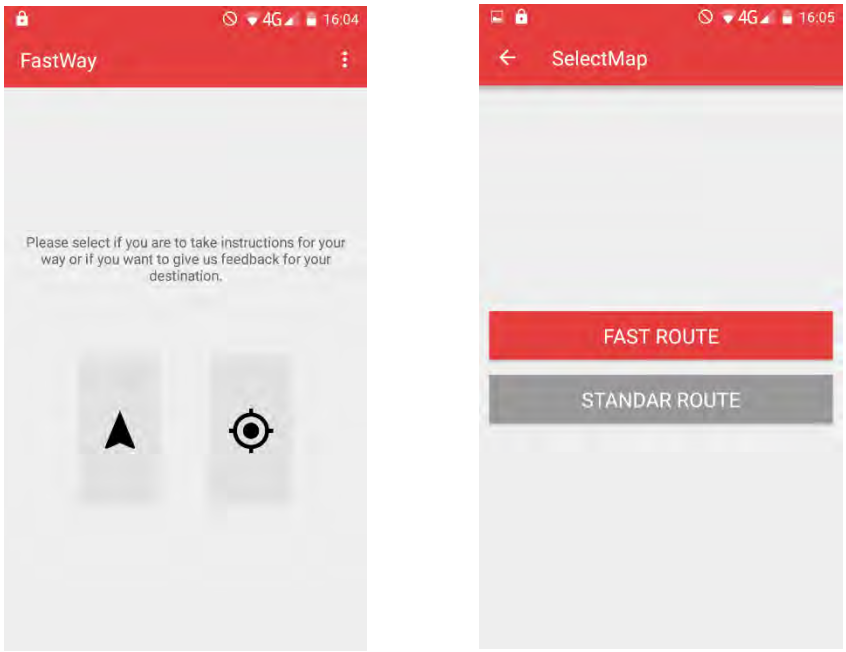
Η εφαρμογή κατά την εκκίνηση της, ελέγχει την περίπτωση που ο χρήστης έχει ξανασυνδεθεί σε αύτη, έτσι ώστε να του εμφανιστούν κατευθείαν οι βασικές επιλογές της εφαρμογής μας. Ωστόσο εάν δεν έχει ξανασυνδεθεί ή μετά την τελευταία του είσοδο σε αύτη έχει αποσυνδεθεί, τότε θα εμφανίζεται μια φόρμα η οποία θα του ζητάει να εισάγει το username και το password του για να συνδεθεί σε αύτη. Στην περίπτωση που εισάγει λάθος στοιχεία ή δεν είναι συνδεδεμένος στο internet τότε θα του εμφανιστούν κατάλληλα μηνύματα για να τον ενημερώσουν σχετικά με τα σφάλματα που δημιουργήθηκαν. Άμα τυχόν τώρα ο χρήστης δεν έχει λογαριασμό τότε του δίνεται η δυνατότητα πατώντας το κουμπί Register να του εμφανιστεί μια φόρμα η οποία θα του ζητάει να εισάγει ένα username και ένα password με τα οποία ο χρήστης θα μπορεί να εισέρχεται με αυτά στην εφαρμογή. Άμα δημιουργηθούν σφάλματα είτε κατά την αποθήκευση των στοιχείων του χρήστη στην βάση δεδομένων όπως ίδιο όνομα είτε στο user interface της ίδιας της εφαρμογής όπως μη συμπλήρωση πεδίων, λιγότερη από 6 χαρακτήρες στον κωδικό και μη

ύπαρξη σύνδεσης στο διαδίκτυο εμφανίζονται κατάλληλα μηνύματα στο user interface για να ενημερωθεί ο χρήστης και να πράξει αναλόγως.



Εικόνα 6.3: Register User Interface και μηνύματα σφάλματων

Η βασική οθόνη της εφαρμογής μας δίνει στο χρήστη δυο επιλογές. Η πρώτη επιλογή αφορά την βασική λειτουργία της εφαρμογής η οποία έχει σχέση με το αν θέλει να λάβει οδηγίες σχετικά με την γρηγορότερη διαδρομή. Η δεύτερη επιλογή είναι άμα θέλει να δώσει πληροφορίες στην εφαρμογή σχετικά με την διάρκεια και από ποιες τοποθεσίες πέρασε για να φτάσει στον τελικό του προορισμό.



Εικόνα 6.4: Βασικές επιλογές εφαρμογής(δεξια)-SelectMap Activity (αριστερα)

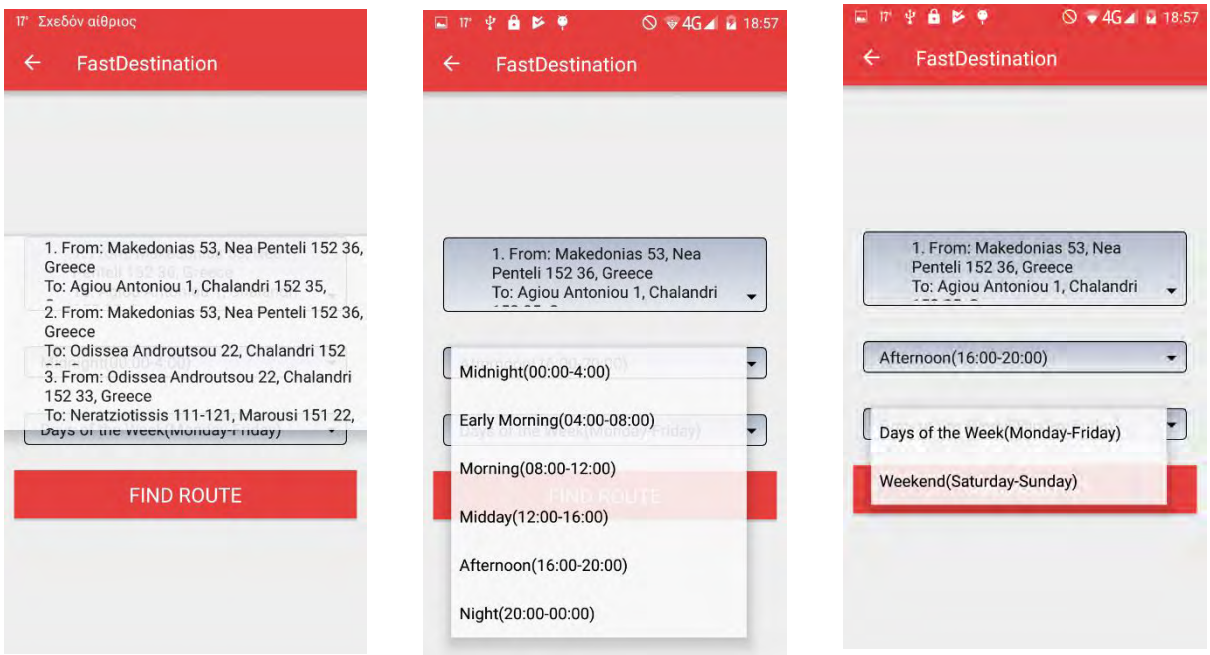
Με το πάτημα του κουμπιού Navigation στην οθόνη του χρήστη, θα εμφανιστούν δυο νέες επιλογές(SelectMap). Το πρώτο κουμπί (Fast Route) εμφανίζει στην οθόνη του τους προορισμούς

για τους οποίους υπάρχουν διαδρομές στην βάση δεδομένων της εφαρμογής, από τις οποίες μπορούμε να αντλήσουμε πληροφορίες για να βρεθεί η συντομότερη διαδρομή. Οι προορισμοί στην οθόνη αυτή, εμφανίζονται με την μορφή Spinner, με το χρήστη να διαλέγει μια από τις διαθέσιμες. Επιπλέον υπάρχουν άλλα 2 spinner με το πρώτο να αφορά την ώρα της ημέρας της οποίας ο χρήστης θέλει να λάβει διαδρομή, ενώ το δεύτερο σχετίζεται με την ημέρα στην οποία επιθυμεί να λάβει τις πληροφορίες για την διαδρομή. Όσον αφορά την επιλογή της ώρα της ημέρας, αύτη ομαδοποιείται με τον τρόπο που φαίνεται παρακάτω:

- Midnight (00:00-04:00)
- Early Morning (04:00-08:00)
- Morning (08:00-12:00)
- Midday (12:00-16:00)
- Afternoon (16:00-20:00)
- Night (20:00-00:00)

Για τις ημέρες της εβδομάδας, οι επιλογές που έχει ο χρήστης είναι 2:

- Days of the Week (Monday – Friday) (καθημερινές)
- Weekends(Saturday-Sunday) (Σαββατοκύριακο)

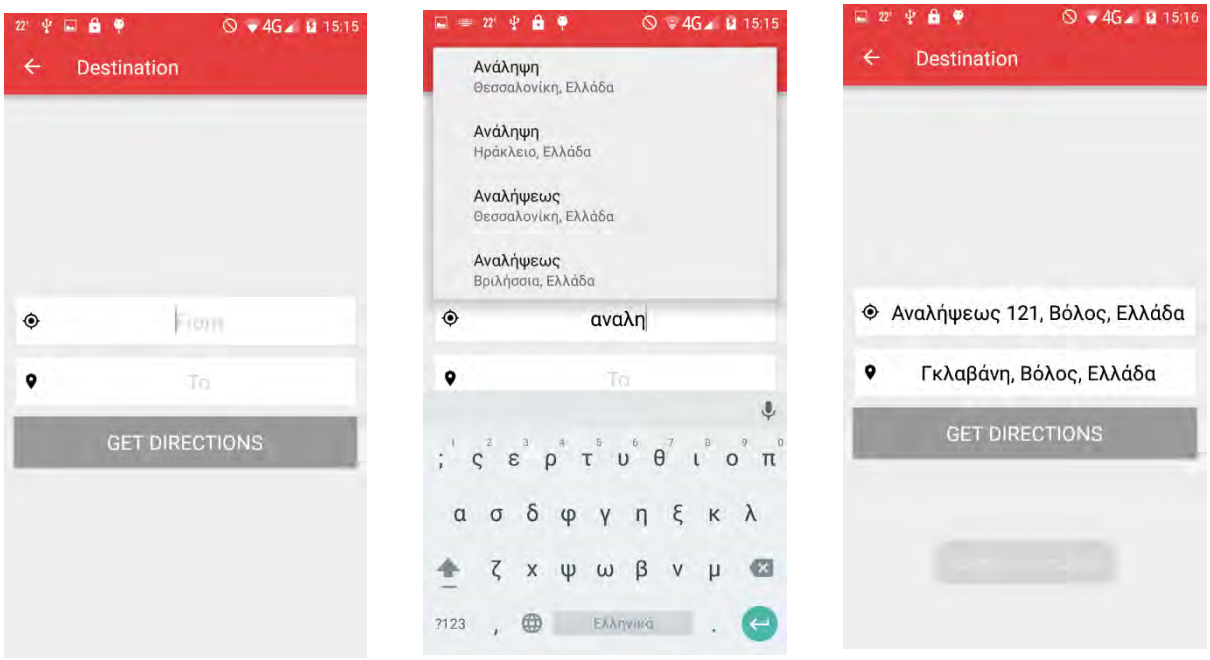


Εικόνα 6.5: FastDestination User interface

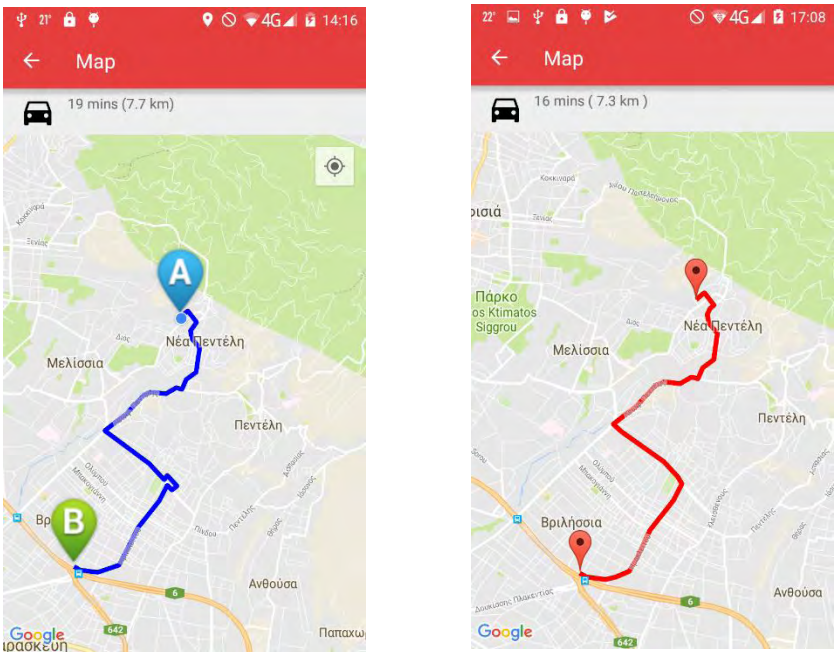
Όταν συμπληρώσει ο χρήστης τις επιλογές του, τότε με το πάτημα του κουμπιού Find Route, θα εμφανιστεί στην οθόνη ο χάρτης της Google και σε αυτόν θα υπάρχει σχεδιασμένη η συντομότερη διαδρομή. Επιπλέον στην οθόνη θα εμφανίζονται τα χιλιόμετρα και η χρονική διάρκεια η οποία χρειάζεται ο χρήστης για να φτάσει στο επιθυμητό προορισμό. Σε περίπτωση που υπάρξει κάποιο πρόβλημα με τον αλγόριθμο, δηλαδή αδυναμία εμφάνισης αποτελέσματος με βάση τα δεδομένα που στάλθηκαν τότε εμφανίζεται κατάλληλο μήνυμα στον χρήστη δίνοντάς του τις επιλογές να εισάγει νέα στοιχεία ή να πάει στην οθόνη Destination.

Στην περίπτωση που δεν επιθυμεί ο χρήστης να βρει την συντομότερη διαδρομή και απλά επιθυμεί να δει την διαδρομή για τον προορισμό του, τότε μπορεί απλά να πατήσει το κουμπί (Standar Route) στην οθόνη SelectMap. Με το πάτημα αυτού του κουμπιού θα εμφανιστούν δυο Autocomplete Text View, τα οποία θα πρέπει να συμπληρωθούν. Αρχικά στο πρώτο πεδίο θα πρέπει να συμπληρωθεί η αφετηρία της διαδρομής ενώ ο προορισμό του χρήστη. Τα πεδία αυτά

έχουν δημιουργηθεί με τέτοιο τρόπο έτσι ώστε καθώς συμπληρώνονται από τον χρήστη να εμφανίζονται διευθύνσεις στην οθόνη οι οποίες θα διευκολύνουν την επιλογή της επιθυμητής διεύθυνσης. Μόλις ο χρήστης συμπληρώσει τα πεδία και στην συνέχεια πατήσει το κουμπί Get Directions εμφανίζεται, στην οθόνη του χρήστη ο χάρτης της Google με σχεδιασμένη την βασική διαδρομή που συνιστά η Google για την συγκεκριμένη διαδρομή. Επιπλέον και σε αυτή την οθόνη εμφανίζεται ο χρόνος και η απόσταση που χρειάζεται για αύτη την διαδρομή.



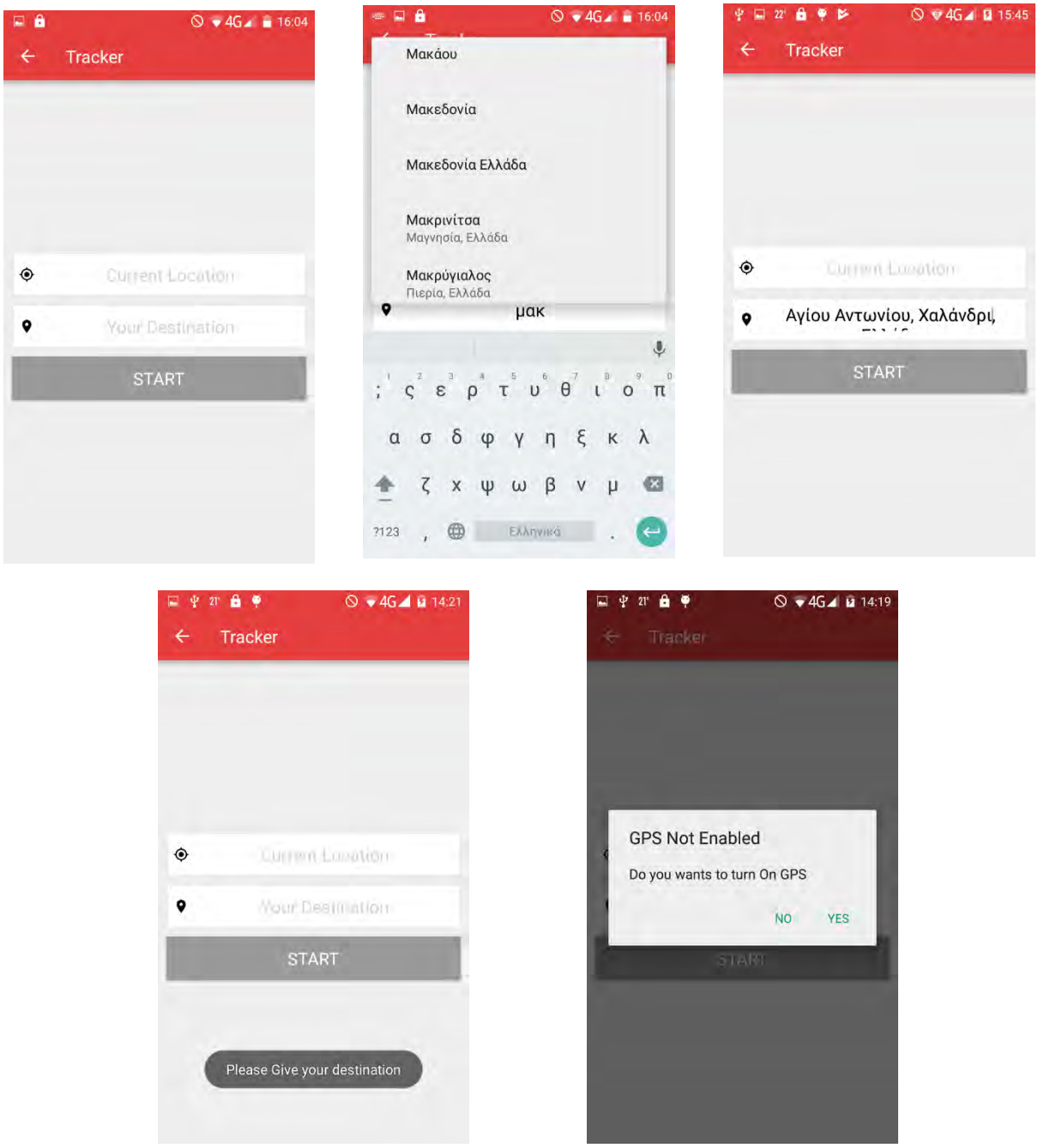
Εικόνα 6.6: Destination User Interface



Εικόνα 6.7: FastMap (Αριστερά) – StandarMap (δεξιά)

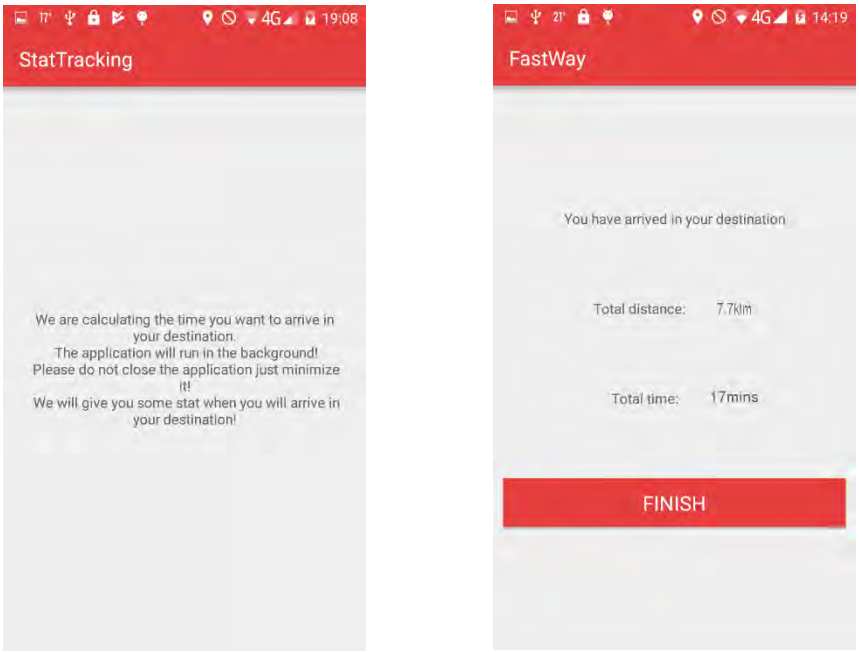
Επιστρέφοντας στην βασική οθόνη της εφαρμογής, ο χρήστης έκτος από το κουμπί Navigation έχει την δυνατότητα να επιλέξει το κουμπί MyLocation. Με το κλικ αυτού του πλήκτρου, θα εμφανιστούν δύο πεδία. Στο δεύτερο πεδίο το οποίο είναι ένα AutoCompleteText View, ο χρήστης θα χρειαστεί να πληκτρολογήσει τον τελικό του προορισμό. Και σε αυτό το πεδίο, καθώς ο χρήστης πληκτρολογεί θα του εμφανίζονται διευθύνσεις που είναι παρεμφερείς με τις δικές τις λέξεις που πληκτρολογεί. Μετά την συμπλήρωση αυτού του, θα υπάρχει η δυνατότητα στο να πατήσει το κουμπί Start. Άμα τυχόν πατήσει το κουμπί δίχως να έχει συμπληρώσει το πεδίο ή έχει απενεργοποιημένο το Gps τότε εμφανίζεται κατάλληλο μήνυμα σφάλματος για να ενημερωθεί ο χρήστης.

Μετά το κλικ του Start η εφαρμογή μας εμφανίζει μια οθόνη η οποία περιέχει ένα μήνυμα προς τον χρήστη να μην κλείσει την εφαρμογή γιατί καταγράφεται η τρέχουσα θέση του . Η οθόνη αύτη



Εικόνα 6.8: Tracker User Interface κα Μηνύματα Σφάλματος

παραμένει ενεργή στην οθόνη του ως ότου ο χρήστης φτάσει στον προορισμό του. Μόλις ο χρήστης φτάσει στον προορισμό του τότε η εφαρμογή μεταβαίνει σε μια οθόνη η οποία δίνει κάποια στατιστικά στοιχεία σχετικά με την διαδρομή, τα οποία σχετίζονται με την απόσταση που διάνυσε και την χρονική διάρκεια που χρειάστηκε για να ολοκληρώσει αύτη την διαδρομή, και ένα κουμπί το οποίο επιστρέφει τον χρήστη στην αρχική οθόνη.
Τέλος πρέπει να τονιστεί ότι σε όλα τα κουμπιά τα οποία χρειάζεται να σταλούν στοιχεία στον server υπάρχει κατάλληλος έλεγχος για να εμφανιστεί μήνυμα ενεργοποιήσεις άμα τυχόν δεν υπάρχει σύνδεση στο διαδίκτυο.



Εικόνα 6.9: StatTracking User Inderface(αριστέρα)- TrackingFinish User Interface (δεξιά)

Εικόνα 6.10:Μήνυμα σφάλματος Internet

6.1.2. Κλάση NetConnection

Για τις κλήσεις του server, δημιουργήθηκε μια ξεχωριστή κλάση με όνομα NetConnection, η οποία κάθε φόρα που χρειάζεται, καλείται για να μπορέσει να γίνει η επικοινωνία με τον server. Συνολικά μέσα σε αύτη την κλάση υπάρχουν 8 συναρτήσεις όπου η κάθε μία επιτελεί διαφορετική λειτουργία. Οι συναρτήσεις που υπάρχουν μέσα στην κλάση αυτή είναι οι εξής:

- existUser: συνάρτηση η οποία κάνει κλήση στον server για τον έλεγχο της ύπαρξης χρήστη
- newUser: συνάρτηση η οποία κάνει κλήση στον server για την εισαγωγή νέου χρήστη στην βάση δεδομένων
- sendLoc: συνάρτηση η οποία κάνει κλήση στον server για την εισαγωγή του νέου τμήματος στην βάση δεδομένων
- sendDestination: συνάρτηση η οποία κάνει κλήση στον server για να εισάγει μια καινούργια διαδρομή κατά την λειτουργία του StatTracking
- getDistanceOnRoad: συνάρτηση η οποία κάνει κλήση στον server για να πάρουμε την απόσταση μεταξύ δύο σημείων
- findDestination: συνάρτηση η οποία κάνει κλήση στον server για να πάρουμε όλες τις διαδρομές που υπάρχουν στην βάση δεδομένων
- getDirections: συνάρτηση η οποία κάνει κλήση στον server για να πάρει από τον server την συντομότερη διαδρομή. Ουσιαστικά είναι η σημαντικότερη συνάρτηση του συστήματος μας καθώς μέσω αυτής γίνεται κλήση του αλγορίθμου FastWay.
- isNetworkAvailable : συνάρτηση η οποία ελέγχει για την ύπαρξη σύνδεσης της συσκευής στο διαδίκτυο΄

```
public void sendLoc(final String startPoint, String endPoint,String ourtime, final Context ctx){
    data=new LocationData(ctx);
    Log.d("Send Location:",startPoint+" endPoint:" + endPoint);
    String url=serverIP+"car?startpos="+startPoint+"&finalpos="+endPoint+"&ourtime="+ourtime+"&id="+data.id() ;
    Log.d("Send mesage",url);

    JsonObjectRequest jsObjRequest = new JsonObjectRequest
            (Request.Method.GET, url, null, (response) → {
                    try {
                        Log.d("sendLoc",response.toString());
                        String distance =response.getString("distance");
                        String time =response.getString("time");
                        double dis=Double.parseDouble(distance);
                        double disPref=Double.parseDouble(data.getTotalDistance());
                        int timeInt=Integer.parseInt(time);
                        Log.d("Response", "distance:"+distance+ "time:"+ time);
                        data.total(dis+disPref+"",timeInt+data.getTotalTime());
                        Log.d("DATA", "distance:"+data.getTotalDistance()+ "time:"+ data.getTotalTime());


                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

            },
                    (error) → { Log.e("Volley", "Error"); });

    RequestQueue rQueue = Volley.newRequestQueue(ctx);
    rQueue.add(jsObjRequest);

}
```

Εικόνα 6.11: Ενδεικτικός κώδικας κλήσης για την συνάρτηση sendLoc

Η επικοινωνία μέσω της εφαρμογής και του server γίνεται μέσω της βιβλιοθήκης Volley. Η απάντηση που θα πάρουμε είναι της μορφής JSON και γίνεται κατάλληλος χειρισμός. Ενδεικτικό κομμάτι κώδικα με την κλήση συνάρτησης ακολουθάει στην Εικόνα 6.10.

6.1.3. Κλάση GpsStatistics

Η κλάση GpsStatistics είναι μια κλάση η οποία λειτουργεί σαν ένα service. Με την έννοια Service εννοούμε ένα συστατικό της εφαρμογής το οποίο μπορεί να εκτελεί μακροσκελείς λειτουργίες στο background και δεν παρέχεται user interface.

Στην παρούσα εφαρμογή η κλάση GpsStatistics ξεκινάει από την activity StatTracking με την εντολή της Εικόνας 6.11

```
trackData=new LocationData(this);
Intent intent = new Intent(getApplicationContext(), GpsStatistics.class);
startService(intent);
fn_permission();
```

Εικόνα 6.12: Εκκίνηση GpsStatistics Service

```
private void newLocation(Location location) {
    tEnd = System.currentTimeMillis();
    latitude = location.getLatitude();
    longitude = location.getLongitude();
    DecimalFormatSymbols symbols = DecimalFormatSymbols.getInstance();
    symbols.setDecimalSeparator('.');
    String stopPoint = latitude + "," + longitude;
    Log.d("StopPoint","StopPoint "+stopPoint+ "TrackPoint"+ trackData.trackPoint());
    String[] from = trackData.end().split(",");
    double latitudeEnd = Double.parseDouble(from[0]);
    double longitudeEnd = Double.parseDouble(from[1]);
    Location locationEnd = new Location("endPoint");
    locationEnd.setLatitude(latitudeEnd);
    locationEnd.setLongitude(longitudeEnd);
    con.getDistanceOnRoad(latitude, longitude, latitudeEnd, longitudeEnd, this);
    if(firstTime){
        Log.d("Distance",distance);
        if (Double.parseDouble(distance)<=0.1) {
            con.sendLoc(trackData.trackPoint(), stopPoint, TimeUnit.MILLISECONDS.toMinutes(tEnd - tStart) + "", this);
            mTimer.cancel();
            Intent dialogIntent = new Intent(this, TrackerFinish.class);
            dialogIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(dialogIntent);

        } else {
            if (tEnd - tStart > 120000) {
                Log.d("MIDPOINT", "PointA" + trackData.trackPoint() + "PointB" + stopPoint);
                midpoint=trackData.trackPoint();
                trackData.tracking(stopPoint);
                con.sendLoc(midpoint, stopPoint, TimeUnit.MILLISECONDS.toMinutes(tEnd - tStart) + "", this);
                tStart = System.currentTimeMillis();
            }
        }
    }
}
```

Εικόνα 6.13: Κώδικας για newLocation

Το Service αυτό έχει ρυθμιστεί να καλείται ανά 30 δευτερόλεπτα για να λαμβάνει την νέα τοποθεσία του χρήστη. Η τοποθεσία του χρήστη δεν αποστέλλεται συνεχώς αλλά μόνο μετά το πέρας τον 2 λεπτών. Υπάρχει κατάλληλος έλεγχος για να κληθεί η συνάρτηση sendLoc και να ελεγχθεί άμα ο χρήστης έχει φτάσει στον προορισμό του έτσι ώστε να σταματήσει να καλείται το service.

6.1.4. Αποθήκευση λογαριασμού στην συσκευή

Ένα σημαντικό χαρακτηριστικό το οποίο έπρεπε να υλοποιηθεί κατά την ανάπτυξη της εφαρμογής, αφορούσε τον τρόπο με τον οποίο κατά την εκκίνηση της κατά την εκκίνηση της εφαρμογής, δεν θα ήταν απαραίτητη η εισαγωγή στοιχείων από τον χρήστη. Για το λόγο αυτό, δημιουργήσαμε μια κλάση με τον όνομα User, στην οποία όταν ο χρήστης εγγράφεται για πρώτη φορά στην εφαρμογή ή όταν συνδέεται πρώτη φορά με τον λογαριασμό του, τότε θα αποθηκεύονται τα στοιχεία του στην μνήμη του τηλεφώνου. Έτσι κάθε φορά που εκκινείτε η εφαρμογή, ελέγχεται εάν στην μνήμη της υπάρχουν αποθηκευμένα δεδομένα. Στην περίπτωση που υπάρχουν δεδομένα τότε η εφαρμογή εμφανίζει την οθόνη με τις βασικές επιλογές της εφαρμογής και όχι αυτή του Login.

```
if(user.loggedIn(this)){
    findViewById(R.id.loginlayout).setVisibility(View.VISIBLE);
    findViewById(R.id.nologinlayout).setVisibility(View.GONE);

}
else {
    findViewById(R.id.loginlayout).setVisibility(View.GONE);
    findViewById(R.id.nologinlayout).setVisibility(View.VISIBLE);
    login=(Button)findViewById(R.id.login);
    signup=(Button)findViewById(R.id.register);
    bar=(ProgressBar)findViewById(R.id.progress_bar);

}
```

Εικόνα 6.14: Κώδικας για την αλλαγή activity εμφάνισης από το Login στο Action

```
public User(Context c){
    pref = c.getSharedPreferences("user", Context.MODE_PRIVATE);
    editor = pref.edit();
}

public void logIn(String username){
    editor.putString("username",username);
    editor.commit();
}


public boolean loggedIn(Context ctx){

    return pref.contains("username");
}
```

Εικόνα 6.15: Κώδικας για αποθήκευση λογαριασμού στην μνήμη της εφαρμογής

6.1.5. Αποφυγή αποστολής πολλαπλών κλήσεων στο server

Ένα πρόβλημα που εμφανίστηκε κατά τον σχεδιασμό της εφαρμογής, αφορούσε τον τρόπο με τον οποίο θα γινόταν η αποφυγή της αποστολής πολλαπλών κλήσεων με το πάτημα του κουμπιού που έστελνε αιτήσεις προς τον server. Για να λυθεί το πρόβλημα αυτό, η εφαρμογή σχεδιάστηκε με τέτοιο τρόπο έτσι ώστε με το πολλαπλό πάτημα του συγκεκριμένου κουμπιού, να απενεργοποιούνται οι λειτουργείς που παρείχε. Έτσι μέσω του παραπάνω σχεδιασμού, δεν θα υπήρχε η περίπτωση να αποσταλούν πολλαπλές αιτήσεις στον server, οι οποίες θα μπορούσαν να δημιουργήσουν προβλήματα τόσο στην εφαρμογή όσο και στον server. Επιπλέον χρειάστηκε να δημιουργηθεί και αντίστοιχη μέθοδος για την ενεργοποίηση των κουμπιών σε περίπτωση σφαλμάτων.

```
public void disablleButtons(){
    login.setEnabled(false);
    signup.setEnabled(false);
    bar.setVisibility(View.VISIBLE);
}

public void enableButtons(){
    login.setEnabled(true);
    signup.setEnabled(true);
    bar.setVisibility(View.GONE);
}
```

Εικόνα 6.16: Κώδικας ενεργοποίησής-απενεργοποίησης κουμπιών

6.2. FastWay Server

Ο server της εφαρμογής, έχει σχεδιαστεί στο εργαλείο Eclipse, όπως αναφέρθηκε και προηγουμένως ακολουθώντας την αρχιτεκτονική REST. Η υπηρεσία είναι εγκατεστημένη τοπικά και έχει γίνει και διαθέσιμος online. Η βασική του λειτουργία, αφορά την διαμεσολάβηση μεταξύ android εφαρμογής και της βάσης δεδομένων. Σκοπός του server είναι να παίρνει τις HTTP Post αιτήσεις που γίνονται από την εφαρμογή, να τις επεξεργάζεται και να κάνει την κατάλληλη κλήση στην περίπτωση που χρειαστεί στην βάση δεδομένων ή στο Google Maps API. Μετά το πέρας της επεξεργασίας από τον server, το αποτέλεσμα επιστρέφεται στην εφαρμογή σε JSON μορφή.

Ο server έχει 7 βασικές λειτουργίες:

- AllDestination
- CarStat
- Destination
- Distance
- FastestRoute
- Login
- Register

Να σημειωθεί ότι η βασικότερη απ' όλες τις λειτουργίες, είναι η FastestRoute στην οποία βρίσκεται ο αλγόριθμος για την εύρεση της συντομότερης διαδρομής. Τέλος υπάρχει ξεχωριστή κλάση με όνομα DBAction, η οποία περιέχει όλες τις κλήσεις που κάνει ο server με την βάση δεδομένων.

6.2.1. FastestRoute

Αποτελεί την σημαντικότερη λειτουργία που επιτελεί ο server. Η λειτουργία αύτη, καλείται από την εφαρμογή για να βρεθεί η γρηγορότερη διαδρομή. Σε αύτη την κλάση υπάρχουν τόσο κλήσεις προς την βάση δεδομένων όσο και προς το Google Maps API.

Η κλήση της συγκεκριμένης λειτουργίας, θα πρέπει να περιέχει ως παραμέτρους την μέρα για την οποία ενδιαφέρεται ο χρήστης να βρεθεί η διαδρομή, την ώρα της ημέρας και το αναγνωριστικό της διαδρομής.

```
public JSONObject FastestRoute(@QueryParam("time") String time, @QueryParam("day") String day,
        @QueryParam("id") String id) {
```

Εικόνα 6.17: Παράμετροι κλήσης FastestRoute

Η κλάση FastRoute ξεκινάει με την κλήση της συνάρτησης findCor η οποία βρίσκεται στην κλάση DBAction. Μέσα από αύτη την συνάρτηση στέλνεται το αναγνωριστικό της διαδρομής για το οποίο ενδιαφερόμαστε και κάνουμε κλήση στην βάση δεδομένων για να επιστραφεί η αρχική και η τελική θέση αυτής της διαδρομής.

```
public ArrayList<String> findCor(String id) throws Exception {
    coordinates=new ArrayList<>();
    Connection dbConn = null;
    try {
        try {
            dbConn = DBConnention();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        Statement stmt = dbConn.createStatement();
        String query = "SELECT * FROM destinations WHERE id = '" + id + "'";
        ResultSet rs = stmt.executeQuery(query);
        if (rs.next()) {
            // rs.getString(3));
            coordinates.add(rs.getString("startpos"));
            coordinates.add(rs.getString("finalpos"));
        }
    } catch (SQLException sqle) {
        throw sqle;
    } catch (Exception e) {
        // TODO Auto-generated catch block
        if (dbConn != null) {
            dbConn.close();
        }
        throw e;
    } finally {
        if (dbConn != null) {
            dbConn.close();
        }
    }
    return coordinates;
}
```

Εικόνα 6.18: Συνάρτηση για την εύρεση της αρχικής και τελικής θέσης της διαδρομής

Αφού μάθουμε την αρχική και τελική θέση, τότε μπορούμε να καλέσουμε την συνάρτηση η οποία ψάχνει να βρει για την γρηγορότερη διαδρομή. Η συνάρτηση αυτή ψάχνει στην βάση δεδομένων και συγκεκριμένα στο table carstats για να βρει όλες εκείνες τις διαδρομές που ξεκινάνε με την αρχική θέση της διαδρομής μας. Αφού βρεθούν και αποθηκευτούν όλα τα στοιχεία που μας ενδιαφέρουν σε κατάλληλες Arraylist, υπάρχει έλεγχος για το εάν βρέθηκαν στοιχεία στην βάση δεδομένων με αυτό το στοιχείο ως αφετηρία και στην περίπτωση που αυτό δεν ισχύει τότε η διαδρομή σημαίνει ότι έφτασε στο πέρας της. Εάν όμως βρεθούν στοιχεία στην βάση δεδομένων, τότε καλείται η συνάρτηση findMax η οποία έχει ως σκοπό την εύρεση των τριών διαδρομών, οι οποίες έχουν διανύσει την μεγαλύτερη απόσταση μέσα σε δυο λεπτά, με το χρονικό διάστημα αυτό να είναι ο καθορισμένος χρόνος για την καταγραφή των θέσεων του χρήστη κατά την

διάρκεια της διαδρομής. Σε αυτό το σημείο θα πρέπει να τονιστεί ότι στην αρχή της συνάρτησης έχει επιλεχθεί τυχαία ένας αριθμός μεταξύ του ένα έως το τρία ο οποίος αφορά ποια από τις τρεις διαδρομές θα επιλεχθούν. Ο λόγος για την παραπάνω λειτουργία επιλέχθηκε με βάση το ότι επιθυμούμε η εφαρμογή να μην στέλνει όλους τους χρήστες από έναν συγκεκριμένο δρόμο το οποίο θα έχει ως αποτέλεσμα αντί να βοηθήσει στην κίνηση στον δρόμο να προκαλέσει μεγαλύτερη συμφόρηση. Εφόσον αποθηκευτούν στην λίστα route που περιέχει όλα τα σημεία της διαδρομής που θα επιστρέφουν στην κύρια συνάρτηση μας το σημείο εκκίνησης αλλάζει και παίρνει αυτό του σημείου τερματισμού της τελευταίας διαδρομής.

Όταν ολοκληρωθεί αύτη η διαδικασία και η λίστα με τα σημεία επιστρέψει στην κύρια συνάρτηση τότε τα σημεία αυτά τοποθετούνται στην κλήση του Google Maps API η οποία επιστέφει την διαδρομή μεταξύ των σημείων που θέλουμε.

Αφού διαβάσουμε την διαδρομή που μας δίνει η Google μπορούμε να την επιστρέψουμε στην εφαρμογή η οποία την χειρίζεται κατάλληλα.

```
public ArrayList<String> selectRoute(String startPos, String finalPos, String time, String day, String id)
        throws SQLException, Exception {

    route = new ArrayList<>();
    Random rand = new Random();
    int max=2;
    int min=0;
    int randomNum = rand.nextInt((max - min) + 1) + min;
    System.out.println("Select Route Algorithm!!!");


    String point = startPos;
    Connection dbConn = null;
    int i = 0;
    dbConn = DBConnention();
    Statement stmt = dbConn.createStatement();
    while (true) {
        try {

            waypoints = new ArrayList<>();
            duration = new ArrayList<>();
            distance = new ArrayList<>();
            sortedwaypoints = new ArrayList<>();
            sorteddurations = new ArrayList<>();
            sorteddistance = new ArrayList<>();
            String query = "SELECT * FROM carstats WHERE id='" + id + "'AND daytime='" + time + "' AND day='" + day
                    + "' AND startpos='" + point + "'";
            System.out.println("Sql = " + query);
            ResultSet rs = stmt.executeQuery(query);
            while (rs.next()) {
                waypoints.add(rs.getString("finalpos"));
                duration.add(rs.getString("ourtime"));
                distance.add(rs.getString("distance"));
            }
            if(waypoints.size()==0){
                break;
            }



            findMax(waypoints.size());
            if(sortedwaypoints.size()>randomNum){
                route.add(sortedwaypoints.get(randomNum));
                point=sortedwaypoints.get(randomNum).toString();
            }
            else if(sortedwaypoints.size()>randomNum-1){
                route.add(sortedwaypoints.get(randomNum-1));
                point=sortedwaypoints.get(randomNum-1).toString();
            }
            else{
                route.add(sortedwaypoints.get(0));
                point=sortedwaypoints.get(0).toString();
            }
            if(startPos.equals(point)){
                return null;
            }
            //System.out.println(point+ " randNumber =" + randomNum);
        } catch (Exception e) {
```

Εικόνα 6.18 Αλγόριθμος FastWay

6.2.2. CarStats

Το CarStats είναι άλλη μια σημαντική λειτουργία που επιτελεί ο server.Είναι αύτη η οποία καλείται για να εισάγει στην βάση δεδομένων τα τμήματα των διαδρομών που στέλνονται από την εφαρμογή.

Η λειτουργία αύτη δέχεται ως παράμετρούς την αρχική και τελική θέση την ώρα που χρειάστηκε για την συγκεκριμένη απόσταση και το αναγνωριστικό της συνολικής διαδρομής.

```
public JSONObject doCarStats( @QueryParam("startpos") String startpos, @QueryParam("finalpos") String finalpos,@QueryParam("ourtime") String ourtime,@QueryParam("id") String id){
    System.out.println("Point A: "+ startpos+ "Point B : " + finalpos);
```

Πριν αποσταλούν τα δεδομένα στην βάση δεδομένων η υπηρεσία αύτη χρειάζεται να βρει την κατηγορία στην οποία ανοίγει η ημέρα στην οποία καλούνται τα δεδομένά και σε ποια κατηγορία από τις ώρες τις ημέρας ανήκει η ώρα στην οποία κλήθηκε η εφαρμογή.

Αφού συλλέξουμε όλα τα δεδομένα η συνάρτηση μας είναι σε θέση να καλέσει την συνάρτηση insertCarData από την DBAction μέσα από την οποία αποθηκεύονταν τα στοιχεία στο table carstats.

Η εντολή SQL η οποία αφορά την αποθήκευση των δεδομένων είναι αύτη που βρίσκεται στην Εικόνα 6.18.

```
String query = "INSERT into carstats(startpos,finalpos,idealtime,ourtime,daytime,day,distance,id) values('"
        + start + "'," + "'" + finalpos + "','" + duration + "','" + ourTime + "','" + time + "','"
        + dayofweek + "','" + distance + "','" + id + "')";
```

Εικόνα 6.19: Εντολή SQL για αποθήκευση δεδεμενων

6.2.3. Υπόλοιπες Λειτουργίες

Στον server μας υπάρχουν και λειτουργίες οι οποίες δεν είναι τόσο σημαντικές όσο οι προηγούμενες αλλά είναι και αυτές πολύ σημαντικές για την σωστή λειτουργία τόσο της εφαρμογής όσο και του server.

Αρχικά έχουμε την κλήση AllDestination είναι η λειτουργία αύτη η οποία όταν καλείται ο server επιστρέφει όλες τις διαδρομές οι οποίες βρίσκονται στην βάση δεδομένων. Για την κλήση του δεν χρειάζεται να υπάρχει κάποια παράμετρος και επιστρέφει όλες τις διαδρομές που υπάρχουν στην βάση δεδομένων.

Η συνάρτηση με την οποία γίνεται η επικοινωνία μέσω server και βάσης δεδομένων είναι η returnDestinations.

```
}
Statement stmt = dbConn.createStatement();
String query = "SELECT * FROM destinations";
// System.out.println(query);
ResultSet rs = stmt.executeQuery(query);
```

Εικόνα 6.20: Εντολή SQL για επιστροφή όλων των προορισμών

Η κλήση Destination είναι η λειτουργία αύτη που έχει ως ρόλο να εισάγει τις διαδρομές στην βάση δεδομένων. Οι παράμετροι που χρειάζονται είναι η αρχική και τελική θέση της διαδρομής μας και επιστρέφει ένα αναγνωριστικό το οποίο σχετίζεται με την θέση που εγγράφτηκε στην βάση δεδομένων. Η συνάρτηση στο DBAction που συνδέει το server είναι το insertDestination. Η

συνάρτηση αύτη πριν εισάγει την διαδρομή χρειάζεται πρώτα να ελεγξει άμα αύτη η διαδρομή έχει ξανά εισαχθεί στην βάση έτσι ώστε να μην υπάρχουν δίπλες εγγραφές.

Με την λειτουργία Distance ο server επιστρέφει στην εφαρμογή την απόσταση μεταξύ δύο σημείων. Είναι η μόνη λειτουργία η οποία ο server δεν χρειάζεται να επικοινωνήσει με την βάση δεδομένων. Η μόνη επικοινωνία που επιτελεί είναι αύτη με το Google Maps API η οποία δίνει την

```
rs = stmt.executeQuery("SELECT id FROM destinations");
// System.out.println(rs.next());

while (rs.next()) {

    i++;
}
rs = stmt.executeQuery("SELECT * FROM destinations WHERE startpos ='" + start + "'");
// System.out.println(rs.next());
while (rs.next()) {

    if (rs.getString("finalpos").equals(finalpos)) {
        //System.out.println(rs.getInt("id"));
        i = rs.getInt("id");
        destinationexist = false;
        break;
    }

}
if (destinationexist) {
    int records = stmt.executeUpdate("INSERT into destinations(startpos, finalpos,id) values('" + start
            + "','" + finalpos + "','" + i + "')");
}
```

Εικόνα 6.21: Κώδικας για έλεγχο μη εισαγωγής διπλών διαδρομών και Εισαγωγής νέας

απόσταση. Οι παράμετροι που χρειαζόμαστε είναι η αρχική και η τελική θέση της διαδρομής και επιστρέφεται η απόσταση μεταξύ τον δύο αυτών σημείων.

Τέλος υπάρχουν και οι λειτουργίες Login και Register. Είναι οι λειτουργίες αυτές οι οποίες σχετίζονται με τον έλεγχο ύπαρξης του χρήστη στην βάση δεδομένων και εγγραφή του χρήστη στην βάση δεδομένων. Οι δύο αυτές λειτουργίες χρειάζονται σαν παραμέτρους για να κληθούν το username του χρήστη και το password και επιστρέφουν κατάλληλα μηνύματα τα οποία τα διαχειρίζεται η εφαρμογή. Η λειτουργία Login επικοινωνεί με την βάση δεδομένων μέσω της συνάρτησης checkLogin και η Register με την συνάρτηση newUser.

Η επικοινωνία του Login και της βάσης μέσω της checkLogin είναι πιο περίπλοκη διότι πρώτα θα χρειαστεί να υπάρξει έλεγχος για το αν το όνομα του χρήστη υπάρχει ήδη στην βάση έτσι ώστε να ενημερωθεί ό χρήστης.

```
public boolean newUser(String uname, String pwd) throws SQLException, Exception {
    boolean insertStatus = false;
    Connection dbConn = null;
    try {
        try {
            dbConn = DBConnention();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        Statement stmt = dbConn.createStatement();
        String query = "SELECT * FROM account WHERE username = '" + uname + "'";
        // System.out.println(query);
        ResultSet rs = stmt.executeQuery(query);
        if (rs.next()) {
            insertStatus = false;
        } else {

            // System.out.println(rs.getString(1) + rs.getString(2) +
            // rs.getString(3));
            query = "INSERT into account(username, password) values('" + uname + "','" + pwd + "')";
            // System.out.println(query);
            int records = stmt.executeUpdate(query);
            // System.out.println(records);
            // When record is successfully inserted
            if (records > 0) {
                insertStatus = true;
            }
        }
```

Εικόνα 6.22: Κώδικας για εγγραφή χρήστη στην βάση δεδομένων

```
public boolean checkLogin(String uname, String pwd) throws Exception {
    boolean isUserAvailable = false;
    Connection dbConn = null;
    try {
        try {
            dbConn = DBConnention();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        Statement stmt = dbConn.createStatement();
        String query = "SELECT * FROM account WHERE username = '" + uname + "' AND password=" + "'" + pwd + "'";
        // System.out.println(query);
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            // System.out.println(rs.getString(1) + rs.getString(2) +
            // rs.getString(3));
            isUserAvailable = true;
        }
```

Εικόνα 6.23: Κώδικας για έλεγχο ύπαρξη λογαριασμού στην βάση δεδομένων

Κεφάλαιο 7

Επίλογος

7.1.Δοκιμές Εφαρμογής

7.1.1. Συσκευές Δοκιμής

Για τον έλεγχο της σωστής λειτουργίας της εφαρμογής, πραγματοποιήθηκαν δοκιμές σε 2 συσκευές android καθώς και στην εικονική μηχανή του Android Studio. Να σημειωθεί ότι εικονική μηχανή, ήταν πάρα πολύ χρήσιμη ειδικά στα πρώτα στάδια της σχεδίασης της εφαρμογής, επειδή μπορούσαμε να αλλάζουμε τις θέσεις του Gps τοπικά. Επομένως μέσω των παραπάνω εργαλείων, βγάλαμε κάποια συμπεράσματα για την σωστή λειτουργία της εφαρμογής καθώς και την επίλυση προβλημάτων τα οποία θα μπορούσαν να προκύψουν κατά τη λειτουργία της.

Μετά την ολοκλήρωση των δοκιμών στην εφαρμογή, πραγματοποιήθηκαν οι δοκιμές και στις υπόλοιπες συσκευές έτσι ώστε να μπορέσουμε να ελέγξουμε τον βαθμό στον οποίο γίνετε σωστά, η απομακρυσμένη κλήση του server καθώς και αν οι θέσεις του χρήστη αλλάζουν σωστά και δουλεύουν όπως επιθυμούμε. Κάποια λάθη που υπήρχαν διορθώθηκαν και στο τέλος επιτευχθεί το επιθυμητό αποτέλεσμα.

Μοντέλο	Ulefone Paris	Samsung S4 Mini	Nexus 5(AVD)
Έκδοση Android	5.1(Lollipop)	4.4.2 (KitKat)	6.0(Marshmallow)
Επεξεργαστής	Octa-core 1.3GHz	Dual-core 1.7 GHz	Dual-Core
Ram	2GB	1.5 GB	1.5GB
Μνήμη Συσκευής	16GB	8 GB	100MB
Μέγεθος οθόνης	5.0 inches HD (720*1280 pixels)	4.3 inches (540 * 960 pixels)	-

Εικόνα 7.1: Χαρακτηριστικά συσκευών που χρησιμοποιήθηκαν

7.1.2. Απόδοση εφαρμογής

Η εφαρμογή σχεδιάστηκε με σκοπό να είναι όσο τον δυνατόν πιο αποδοτική και γρήγορη. Μετά την ολοκλήρωση της, καταγράψαμε τον χρόνο που χρειάστηκε να ξεκινήσει καθένα Activity, το οποίο φαίνεται στον Πίνακα 7.1. Μέσω των αποτελεσμάτων που προέκυψαν στον παρακάτω πίνακα, έγινε δυνατή η παρατήρηση της απόδοσης του server καθώς και της διάρκειας που χρειαζόταν η εφαρμογή για να πραγματοποιήσει ένα request-reply του server, πάνω από ένα αξιόπιστο δίκτυο.

Start Activity	Final Activity	Time	Server Request	Server Function
Start Application	MainActivity	283ms	NO	-
MainActivity(NO LOGIN)	Register	230ms	NO	-
Register	MainActivity(LOGIN)	275ms	YES	Register
Logout	MainActivity(NO LOGIN)	175ms	NO	-
MainActivity(NO LOGIN)	MainActivity(LOGIN)	273ms	YES	Login

MainActivity(LOGIN)	SelectMap	210ms	NO	-
SelectMap	FastDestination	308ms	YES	AllDestinations
FastDestination	FastMap	1s474ms	YES	FastestRoute
SelectMap	Destination	365ms	NO	-
Destination	Map	659ms	YES	GoogleMapsAPI
MainActivity	Tracker	241ms	NO	-
Tracker	StatTracking	236ms	YES	Destination
StatTracking	StatTracking	236ms	YES	Distance,CarStats
StatTracking	FinishTracker	236ms	NO	-

Πίνακας 7.1: Χρόνοι Εκτέλεσης Activity

7.1.3. Στοιχεία Βάσης Δεδομένων

Μετά την ολοκλήρωση της υλοποίησης, ακολούθησαν δοκιμές για να ελεγχθεί η καταγραφή των διαδρομών. Επιπλέον ο σκοπός των δοκιμών αυτών, ήταν η δημιουργία μιας μικρής βάσης δεδομένων, η οποία θα παρουσίαζε αν τελικά ο αλγόριθμος FastWay λειτουργεί σωστά. Για το λόγο αυτό, χρειάστηκε να καταγράψουμε διαφορετικές διαδρομές για τρεις προορισμούς, έτσι ώστε να δοκιμάσουμε τον αλγόριθμό μας. Τα σημεία που καταγράφηκαν αποθηκεύτηκαν στην βάση δεδομένων και παρουσιάζονται στον Πίνακα 7.1.

startPos	finishPos	ourtime	Daytime	Day	Distance	ID
38.064200,23.854240	**38.063449,23.857341**	2	Afternoon	Daily	**0.5**	1
38.063449,23.857341	**38.059673,23.857684**	2	Afternoon	Daily	**0.5**	1
38.059673,23.857684	**38.055052,23.855978**	2	Afternoon	Daily	**0.5**	1
38.055052,23.855978	**38.048479,23.854530**	2	Afternoon	Daily	**0.9**	1
38.048479,23.854530	**38.041981,23.852816**	2	Afternoon	Daily	**0.7**	1
38.041981,23.852816	**38.038288,23.851700**	2	Afternoon	Daily	**0.5**	1
38.038288,23.851700	**38.032745,23.847215**	2	Afternoon	Daily	**0.7**	1
38.032745,23.847215	**38.025782,23.842080**	2	Afternoon	Daily	**0.8**	1
38.025782,23.842080	**38.025326,23.834044**	2	Afternoon	Daily	**0.7**	1
38.064200,23.854240	**38.063449,23.857341**	2	Afternoon	Daily	**0.5**	1
38.063449,23.857341	**38.059673,23.857684**	2	Afternoon	Daily	**0.5**	1
38.059673,23.857684	**38.055052,23.855978**	2	Afternoon	Daily	**0.5**	1
38.055052,23.855978	**38.052389,23.850451**	2	Afternoon	Daily	**0.6**	1
38.052389,23.850451	**38.049483,23.843883**	2	Afternoon	Daily	**0.7**	1
38.049483,23.843883	**38.045199,23.841520**	2	Afternoon	Daily	**0.7**	1
38.045199,23.841520	**38.041946,23.847404**	2	Afternoon	Daily	**0.6**	1
38.041946,23.847404	**38.038288,23.851700**	2	Afternoon	Daily	**0.6**	1
38.038288,23.851700	**38.032745,23.847215**	2	Afternoon	Daily	**0.7**	1
38.032745,23.847215	**38.025782,23.842080**	2	Afternoon	Daily	**0.8**	1
38.025782,23.842080	**38.025326,23.834044**	2	Afternoon	Daily	**0.7**	1
38.064200,23.854240	**38.063449,23.857341**	2	Afternoon	Daily	**0.5**	1
38.063449,23.857341	**38.059673,23.857684**	2	Afternoon	Daily	**0.5**	1
38.059673,23.857684	**38.055052,23.855978**	2	Afternoon	Daily	**0.5**	1
38.055052,23.855978	**38.052389,23.850451**	2	Afternoon	Daily	**0.6**	1
38.052389,23.850451	**38.049483,23.843883**	2	Afternoon	Daily	**0.7**	1

38.049483,23.843883	**38.043735,23.837902**	2	Afternoon	Daily	**0.9**	1
38.043735,23.837902	**38.039037,23.838782**	2	Afternoon	Daily	**0.7**	1
38.039037,23.838782	**38.033882,23.840016**	2	Afternoon	Daily	**0.8**	1
38.033882,23.840016	**38.029226,23.836851**	2	Afternoon	Daily	**0.5**	1
38.029226,23.836851	**38.025829,23.834029**	2	Afternoon	Daily	**0.4**	1
38.0546,23.843348	**38.059767,23.850573**	2	Nigth	Daily	**1**	2
38.064167,23.854261	**38.059768,23.850575**	2	Afternoon	Daily	**0.7**	2
38.059768,23.850575	**38.054600,23.843349**	2	Afternoon	Daily	**0.9**	2
38.054600,23.843349	**38.057117,23.838465**	2	Afternoon	Daily	**0.5**	2
38.057117,23.838465	**38.051888,23.838658**	2	Afternoon	Daily	**0.6**	2
38.051888,23.838658	**38.049571,23.836025**	2	Afternoon	Daily	**0.4**	2
38.049571,23.836025	**38.045580,23.831667**	2	Afternoon	Daily	**0.6**	2
38.045580,23.831667	**38.044095,23.829103**	2	Afternoon	Daily	**0.4**	2
38.044095,23.829103	**38.042387,23.819553**	2	Afternoon	Daily	**0.9**	2
38.042387,23.819553	**38.035432,23.819241**	2	Afternoon	Daily	**0.8**	2
38.035432,23.819241	**38.032127,23.808081**	2	Afternoon	Daily	**1.4**	2
38.032127,23.808081	**38.027986,23.804130**	2	Afternoon	Daily	**0.6**	2
38.064167,23.854261	**38.059768,23.850575**	2	Afternoon	Daily	**0.7**	2
38.059768,23.850575	**38.054600,23.843349**	2	Afternoon	Daily	**0.9**	2
38.054600,23.843349	**38.057117,23.838465**	2	Afternoon	Daily	**0.5**	2
38.057117,23.838465	**38.051888,23.838658**	2	Afternoon	Daily	**0.6**	2
38.051888,23.838658	**38.046320,23.838877**	2	Afternoon	Daily	**0.7**	2
38.046320,23.838877	**38.040070,23.830423**	2	Afternoon	Daily	**1**	2
38.040070,23.830423	**38.038059,23.827716**	2	Afternoon	Daily	**0.3**	2
38.038059,23.827716	**38.034673,23.823466**	2	Afternoon	Daily	**0.5**	2
38.034673,23.823466	**38.033262,23.818836**	2	Afternoon	Daily	**0.5**	2
38.033262,23.818836	**38.030231,23.812878**	2	Afternoon	Daily	**0.7**	2
38.030231,23.812878	**38.027986,23.804130**	2	Afternoon	Daily	**1.1**	2
38.064167,23.854261	**38.059768,23.850575**	2	Afternoon	Daily	**0.7**	2
38.059768,23.850575	**38.054600,23.843349**	2	Afternoon	Daily	**0.9**	2
38.054600,23.843349	**38.057117,23.838465**	2	Afternoon	Daily	**0.5**	2
38.057117,23.838465	**38.051888,23.838658**	2	Afternoon	Daily	**0.6**	2
38.051888,23.838658	**38.046320,23.838877**	2	Afternoon	Daily	**0.7**	2
38.046320,23.838877	**38.040070,23.830423**	2	Afternoon	Daily	**1**	2
38.040070,23.830423	**38.044095,23.829103**	2	Afternoon	Daily	**0.5**	2
38.044095,23.829103	**38.042387,23.819553**	2	Afternoon	Daily	**0.9**	2
38.042387,23.819553	**38.035432,23.819241**	2	Afternoon	Daily	**0.8**	2
38.035432,23.819241	**38.032127,23.808081**	2	Afternoon	Daily	**1.4**	2
38.032127,23.808081	**38.027986,23.804130**	2	Afternoon	Daily	**0.6**	2
38.028346,23.804381	**38.029321,23.801582**	2	Afternoon	Daily	**0.3**	3
38.029321,23.801582	**38.030379,23.797649**	2	Afternoon	Daily	**0.6**	3
38.030379,23.797649	**38.034757,23.790352**	2	Afternoon	Daily	**0.8**	3
38.034757,23.790352	**38.039373,23.792115**	2	Afternoon	Daily	**0.5**	3

38.039373,23.792115	**38.040180,23.792925**	2	Afternoon	Daily	**0.1**	3
38.040180,23.792925	**38.043054,23.789837**	2	Afternoon	Daily	**0.6**	3
38.028346,23.804381	**38.029321,23.801582**	2	Afternoon	Daily	**0.3**	3
38.029321,23.801582	**38.037519,23.807016**	2	Afternoon	Daily	**1.1**	3
38.037519,23.807016	**38.039836,23.802098**	2	Afternoon	Daily	**0.5**	3
38.039836,23.802098	**38.043087,23.801747**	2	Afternoon	Daily	**0.6**	3
38.043087,23.801747	**38.043582,23.797360**	2	Afternoon	Daily	**0.4**	3
38.043582,23.797360	**38.043054,23.789837**	2	Afternoon	Daily	**1.3**	3

Πίνακας 7.1:Στοιχεία Βάσης Δεδομένων CarStats

Μετά την καταγραφή των δεδομένων και με την βοήθεια του Google Maps API, μπορέσαμε να πάρουμε μερικά συμπεράσματα για τις γρηγορότερες διαδρομές με βάση τα σημεία που είχαμε συλλέξει. Οι διαδρομές για τους τρείς προορισμούς, παρουσιάζονται στην Πίνακας 7.2.

startPos	finalPos	ID
38.064200,23.854240	38.025937,23.833552	1
38.064167,23.854261	38.028310,23.804469	2
38.028346,23.804381	38.044573,23.792805	3

Πίνακας 7.2: Διαδρομές Βάσης Δεδομένων Destinations

7.2.Συμπεράσματα

Κατά την διάρκεια της σχεδίασης και υλοποίησης της εφαρμογής, προέκυψαν ορισμένα συμπεράσματα τα οποία φαίνονται παρακάτω.

- Κατά την λειτουργία της εφαρμογής ,παρατηρήθηκε ότι οι μεταβάσεις από το ένα Activity στο άλλο, ήταν πάρα πολύ γρήγορες. Επίσης οι απαντήσεις του server ήταν εξίσου πολύ γρήγορες, με μόνη εξαίρεση την κλήση του αλγορίθμου, η οποία ήταν λίγο πιο αργή.
- Για να μπορέσει η εφαρμογή να καταστεί αποδοτική, χρειάζεται να έχει ένα ικανοποιητικό αριθμό χρηστών, οι οποίοι να μπορούν να προσφέρουν δεδομένα στην εφαρμογή έτσι ώστε να βρίσκονται οι βέλτιστες διαδρομές. Όσο μεγαλύτερη είναι η βάση δεδομένων, τόσο καλύτερα και τα αποτελέσματα που θα δίνει ο αλγόριθμος που έχει δημιουργηθεί.
- Η υλοποίηση μιας εφαρμογής για λειτουργικό σύστημα Android, ήταν σχετικά εύκολή από την στιγμή που υπήρχαν οι κατάλληλες γνώσεις Java. Το πιο περίπλοκο κομμάτι, ήταν η εύρεση του σωστού συνδυασμού των τεχνολογιών, έτσι ώστε να μπορούν να λειτουργούν σωστά.

Παραπομπές

1. FastWay

1.1. MainActivity

```
public class MainActivity extends AppCompatActivity {
   EditText username;
   EditText password;
   private User user;
   private TextView errormessage;
   private Button login;
   private Button signup;
   private ProgressBar bar;
   private NetConnection con;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      user=new User(this);
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      getSupportActionBar().setElevation(0);
      con = new NetConnection();
      con.isNetworkAvailable(this);
      if(user.loggedIn(this)){
         findViewById(R.id.loginlayout).setVisibility(View.VISIBLE);
         findViewById(R.id.nologinlayout).setVisibility(View.GONE);       }
      else {
         findViewById(R.id.loginlayout).setVisibility(View.GONE);
         findViewById(R.id.nologinlayout).setVisibility(View.VISIBLE);
         login=(Button)findViewById(R.id.login);
         signup=(Button)findViewById(R.id.register);
         bar=(ProgressBar)findViewById(R.id.progress_bar);
      }
   }
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the action bar if it is present.
   getMenuInflater().inflate(R.menu.menu_main, menu);
   if (user.loggedIn(this)) {
      ((MenuItem) menu.findItem(R.id.logout)).setVisible(true);
   }
   else{
      ((MenuItem) menu.findItem(R.id.logout)).setVisible(false);
   }
   return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   int id = item.getItemId();
   if (id == R.id.logout) {
      logout();
      return true;
   }
 return super.onOptionsItemSelected(item);
}
public void onLogin(View v){
   username=(EditText)findViewById(R.id.username);
   password=(EditText)findViewById(R.id.password);
   errormessage=(TextView)findViewById(R.id.message);
   if(username.getText().length() < 1){
      errormessage.setText(getResources().getString(R.string.error_name));
      errormessage.setVisibility(View.VISIBLE);
   }
   else if(password.getText().length() < 1){
      errormessage.setText(getResources().getString(R.string.error_password));
      errormessage.setVisibility(View.VISIBLE);
   }
   else{
```

```
            con = new NetConnection();
            disablleButtons();
            InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
            con.existUser(username.getText().toString(),password.getText().toString(),this);
        }
    }
    public void onRegister(View V){
        Intent intent = new Intent(MainActivity.this, Register.class);
        startActivity(intent);
    }
    public void onNavigation(View view){
        Intent intent = new Intent(MainActivity.this, SelectMap.class);
        startActivity(intent);
    }
    public void onLocation(View view){
        con = new NetConnection();
        con.isNetworkAvailable(this);
        Intent intent = new Intent(MainActivity.this, Tracker.class);
        startActivity(intent);
    }
    public void disablleButtons(){
        login.setEnabled(false);
        signup.setEnabled(false);
        bar.setVisibility(View.VISIBLE);
    }
    public void enableButtons(){
        login.setEnabled(true);
        signup.setEnabled(true);
        bar.setVisibility(View.GONE);
    }

    public void logout(){
```

```
            user.logOut();
            Intent intent = new Intent(MainActivity.this, MainActivity.class);
            finish();
            startActivity(intent);
        }
        public void viewStatistics(String username){
            Intent intent = new Intent(MainActivity.this, ViewStats.class);
            startActivity(intent);
        }
    }
```

1.2.FastDestination

```
public class FastDestination extends AppCompatActivity {
    Destinations destination;
    private Spinner spinnertime;
    private Spinner spinnerday;
    private Spinner spinnerdestination;
    private NetConnection con;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_fast_destination);
        spinnertime = (Spinner) findViewById(R.id.time_spinner);
        spinnerday = (Spinner) findViewById(R.id.day_spinner);
        spinnerdestination = (Spinner) findViewById(R.id.destination_spinner);
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
                R.array.time_array, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        ArrayAdapter<CharSequence> adapterDay = ArrayAdapter.createFromResource(this,
                R.array.day_array, android.R.layout.simple_spinner_item);
        adapterDay.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinnertime.setAdapter(adapter);
        spinnerday.setAdapter(adapterDay);
        destination = new Destinations();
```

```
    ArrayAdapter<String> adapterDest =
            new ArrayAdapter<String>(this, android.R.layout.simple_expandable_list_item_1,
destination.getDest());
    adapterDest.setDropDownViewResource(android.R.layout.simple_expandable_list_item_1);
    spinnerdestination.setAdapter(adapterDest);
  }
  public void onfind(View view) {
    con = new NetConnection();
    con.isNetworkAvailable(this);
    disabeButtons();
    String day=spinnerday.getSelectedItem().toString().split("\\(")[0];
    String time=spinnertime.getSelectedItem().toString().split("\\(")[0];
    String dest=spinnerdestination.getSelectedItem().toString().split("\\.")[0];
    if(day.equals("Days of the Week"))
      day="Daily";
    con.getDirections(time,day,dest,this);
  }
  private void disabeButtons() {
    ((Spinner) findViewById(R.id.destination_spinner)).setEnabled(false);
    ((Spinner) findViewById(R.id.day_spinner)).setEnabled(false);
    ((Spinner) findViewById(R.id.time_spinner)).setEnabled(false);
    ((Button) findViewById(R.id.find)).setEnabled(false);
    ((ProgressBar)findViewById(R.id.progress_bar)).setVisibility(View.VISIBLE);
  }
  public void enableButtons() {
    ((Spinner) findViewById(R.id.destination_spinner)).setEnabled(true);
    ((Spinner) findViewById(R.id.day_spinner)).setEnabled(true);
    ((Spinner) findViewById(R.id.time_spinner)).setEnabled(true);
    ((Button) findViewById(R.id.find)).setEnabled(true);
    ((ProgressBar)findViewById(R.id.progress_bar)).setVisibility(View.VISIBLE);
  }
}
```

1.3.FastWayMap

```
public class FastestWayMap extends AppCompatActivity implements OnMapReadyCallback {
    private GoogleMap mMap;
    Destinations destination;
    ArrayList<LatLng> MarkerPoints;
    private List<Marker> originMarkers = new ArrayList<>();
    private List<Marker> destinationMarkers = new ArrayList<>();
    private List<Polyline> polylinePaths = new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_fastest_way_map);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        destination = new Destinations();
    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        LatLng hcmus = new LatLng(10.762963, 106.682394);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(hcmus, 18));
        originMarkers.add(mMap.addMarker(new MarkerOptions()
                .title("Đại học Khoa học tự nhiên")
                .position(hcmus)));
        if (ActivityCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_FINE_LOCATION)
!= PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,android.Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        }
        mMap.setMyLocationEnabled(true);
        try {
            parseJSon(destination.getDirections());
        } catch (JSONException e) {
```

```
            e.printStackTrace();
        }
    }
    private void parseJSon(JSONObject jsonData) throws JSONException {
        List<Route> routes = new ArrayList<>();
        JSONArray jsonRoutes = jsonData.getJSONArray("routes");
        for (int i = 0; i < jsonRoutes.length(); i++) {
                    JSONObject jsonRoute = jsonRoutes.getJSONObject(i);
            Route route = new Route();
            JSONObject overview_polylineJson = jsonRoute.getJSONObject("overview_polyline");
            JSONArray jsonLegs = jsonRoute.getJSONArray("legs");
            JSONObject jsonLeg = jsonLegs.getJSONObject(0);
            JSONObject jsonDistance = jsonLeg.getJSONObject("distance");
            JSONObject jsonDuration = jsonLeg.getJSONObject("duration");
            JSONObject jsonEndLocation = jsonLeg.getJSONObject("end_location");
            JSONObject jsonStartLocation = jsonLeg.getJSONObject("start_location");
            route.setDistance( new Distance(jsonDistance.getString("text"), jsonDistance.getInt("value")));
            route.setDuration(new Duration(jsonDuration.getString("text"), jsonDuration.getInt("value")));
            route.setEndAddress( jsonLeg.getString("end_address"));
            route.setStartAddress (jsonLeg.getString("start_address"));
            route.setStartLocation (new LatLng(jsonStartLocation.getDouble("lat"),
jsonStartLocation.getDouble("lng")));
            route.setEndLocation (new LatLng(jsonEndLocation.getDouble("lat"),
jsonEndLocation.getDouble("lng")));
            route.setPoints (decodePolyLine(overview_polylineJson.getString("points")));
            routes.add(route);
        }
                onDirectionFinderSuccess(routes);
    }
    private List<LatLng> decodePolyLine(final String poly) {
        int len = poly.length();
        int index = 0;
        List<LatLng> decoded = new ArrayList<LatLng>();
        int lat = 0;
```

```
    int lng = 0;
    while (index < len) {
        int b;
        int shift = 0;
        int result = 0;
        do {
            b = poly.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lat += dlat;
        shift = 0;
        result = 0;
        do {
            b = poly.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lng += dlng;
        decoded.add(new LatLng(
                lat / 100000d, lng / 100000d
        ));
    }
    return decoded;
}
public void onDirectionFinderSuccess(List<Route> routes) {
    polylinePaths = new ArrayList<>();
    originMarkers = new ArrayList<>();
    destinationMarkers = new ArrayList<>();
    for (Route route : routes) {
        Log.d("Size",routes.size()+"");
```

```
        Log.d("Direction Finder:",route.getStartLocation().latitude + ","+ route.getStartLocation().longitude
);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(route.getStartLocation(), 16));
        ((TextView) findViewById(R.id.info)).setText(route.getDuration().getText()+ " ("+
route.getDistance().getText()+ ")");
        originMarkers.add(mMap.addMarker(new MarkerOptions()
                .icon(BitmapDescriptorFactory.fromResource(R.drawable.start_blue))
                .title(route.getStartAddress())
                .position(route.getStartLocation())));
        destinationMarkers.add(mMap.addMarker(new MarkerOptions()
                .icon(BitmapDescriptorFactory.fromResource(R.drawable.end_green))
                .title(route.getEndAddress())
                .position(route.getEndLocation())));
        PolylineOptions polylineOptions = new PolylineOptions().
                geodesic(true).
                color(Color.BLUE).
                width(10);
        for (int i = 0; i < route.getPoints().size(); i++)
           polylineOptions.add(route.getPoints().get(i));
        polylinePaths.add(mMap.addPolyline(polylineOptions));
      }
   }
}
```

1.4. GpsStatistics

```
public class GpsStatistics extends Service implements LocationListener {
   boolean isGPSEnable = false;
   boolean isNetworkEnable = false;
   double latitude, longitude;
   LocationManager locationManager;
   Location location;
   private Handler mHandler = new Handler();
   private Timer mTimer = null;
   long notify_interval = 30000;
```

```
public static String str_receiver = "com.example.mitsos_laptop.trafficapp.receiver";
Intent intent;
private boolean firstTime = false;
private long tStart;
private long tEnd;
private NetConnection con;
private String midpoint="37.982157,23.726320";
private String distance = "100.0";
private double totalDistance = 0.0;
private long totalTime = 0;
private LocationData trackData;
public GpsStatistics() {
}
@Nullable
@Override
public IBinder onBind(Intent intent) {
   return null;
}
@Override
public void onCreate() {
   super.onCreate();
   Log.d("Google Service", "on create");
   mTimer = new Timer();
   tStart = System.currentTimeMillis();
   trackData = new LocationData(this);
   //Log.e("initCord",trackData.end());
   con = new NetConnection();
   if(!firstTime)
      midpoint=trackData.begin();
   mTimer.schedule(new TimerTaskToGetLocation(), 5, notify_interval);
   intent = new Intent(str_receiver);
   //fn_getlocation();
}
@Override
```

```
    public void onLocationChanged(Location location) {
    }
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }
    @Override
    public void onProviderEnabled(String provider) {
    }
    @Override
    public void onProviderDisabled(String provider) {

    }
    private void fn_getlocation() {
        locationManager = (LocationManager)
getApplicationContext().getSystemService(LOCATION_SERVICE);
        isGPSEnable = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
        isNetworkEnable = locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
        if (!isGPSEnable && !isNetworkEnable) {
        } else {
            if (isNetworkEnable) {
                location = null;
                if (ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                    return;
                }
                locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 500, 0, this);
                if (locationManager != null) {
                    location = locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                    if (location != null) {
                        newLocation(location);


                    }
                }
```

```
        }
        if (isGPSEnable) {
            location = null;
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 10, this);
            if (locationManager != null) {
                location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                if (location != null) {
                    newLocation(location);
                }
            }
        }
    }
}
private class TimerTaskToGetLocation extends TimerTask {
    @Override
    public void run() {
        mHandler.post(new Runnable() {
            @Override
            public void run() {
                Log.d("Google Service", "timer");
                fn_getlocation();
            }
        });

    }
}
   private void newLocation(Location location) {
    tEnd = System.currentTimeMillis();
    latitude = location.getLatitude();
    longitude = location.getLongitude();
    DecimalFormatSymbols symbols = DecimalFormatSymbols.getInstance();
    symbols.setDecimalSeparator('.');
    DecimalFormat format = new DecimalFormat("#0.######", symbols);
    latitude = Double.valueOf(format.format(latitude));
```

```
    longitude = Double.valueOf(format.format(longitude));
    String stopPoint = latitude + "," + longitude;
    Log.d("StopPoint", "StopPoint " + stopPoint + "TrackPoint" + trackData.trackPoint());
    String[] from = trackData.end().split(",");
    double latitudeEnd = Double.parseDouble(from[0]);
    double longitudeEnd = Double.parseDouble(from[1]);
    Location locationEnd = new Location("endPoint");
    locationEnd.setLatitude(latitudeEnd);
    locationEnd.setLongitude(longitudeEnd);
    con.getDistanceOnRoad(latitude, longitude, latitudeEnd, longitudeEnd, this);
    if (firstTime) {
      Log.d("Distance", distance);
      if (Double.parseDouble(distance) <= 0.1) {
        con.sendLoc(midpoint, stopPoint, TimeUnit.MILLISECONDS.toMinutes(tEnd - tStart) + "", this);
        mTimer.cancel();
        Intent dialogIntent = new Intent(this, TrackerFinish.class);
        dialogIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(dialogIntent);

      } else {
        if (tEnd - tStart > 120000) {
          Log.d("MIDPOINT", "PointA" + trackData.trackPoint() + "PointB" + stopPoint);
          con.sendLoc(midpoint, stopPoint, TimeUnit.MILLISECONDS.toMinutes(tEnd - tStart) + "", this);
          tStart = System.currentTimeMillis();
        }
      }
    }
  }
  public void setDistance(String dis) {
    distance = dis;
    firstTime = true;
  }
  public void setMidpoint(String point) {
    midpoint=point;
```

```
    }
}
```

1.5. NetConnection

```
public class NetConnection {
    protected static final String serverIP = "http://localhost:60000/TrafficServer/";
    private LocationData data;
    private Destinations destination;
    private ArrayList<String> dest=new ArrayList<String>();
    private ArrayList<String> fromList=new ArrayList<String>();
    private ArrayList<String> toList=new ArrayList<String>();
    public void existUser(final String username, String password, final Context ctx){
        String url=serverIP + "login/dologin?username=" + username +"&password="+password;
        Log.d("Send mesage",url);
        final ProgressBar progress = (ProgressBar) ((Activity)ctx).findViewById(R.id.progress_bar);
        final TextView errormsg = (TextView) ((Activity)ctx).findViewById(R.id.message);
        JsonObjectRequest jsObjRequest = new JsonObjectRequest
                (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
                    public void onResponse(JSONObject response) {
                        try {
                            String result =response.getString("login");
                            if(result.equals("true")){
                                User usr=new User(ctx);
                                usr.logIn(username);
                                Toast.makeText(ctx, ctx.getResources().getString(R.string.loggedin),
Toast.LENGTH_SHORT).show();
                                Intent intent = new Intent(((Activity)ctx), MainActivity.class);
                                ((Activity)ctx).finish();
                                ctx.startActivity(intent);
                            }
                            else{
                                ((MainActivity)ctx).enableButtons();
                                progress.setVisibility(View.GONE);
                                errormsg.setText(ctx.getResources().getString(R.string.invalidlogin));
                                errormsg.setVisibility(View.VISIBLE);
```

```
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    Log.e("Volley", "Error");
                }
            });

    RequestQueue rQueue = Volley.newRequestQueue(ctx);
    rQueue.add(jsObjRequest);
}
public void newUser(final String username, String password, final Context ctx){
    String url=serverIP + "register/doregister?username=" + username +"&password="+password;
    Log.d("Send mesage",url);
    final ProgressBar progress = (ProgressBar) ((Activity)ctx).findViewById(R.id.progress_bar);
    final TextView errormsg = (TextView) ((Activity)ctx).findViewById(R.id.message);
    progress.setVisibility(View.VISIBLE);
    JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
            public void onResponse(JSONObject response) {
                try {
                    String result =response.getString("register");
                    if(result.equals("true")){
                        User usr=new User(ctx);
                        usr.logIn(username);
                        Toast.makeText(ctx, ctx.getResources().getString(R.string.registered),
Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(((Activity)ctx), MainActivity.class);
                        ((Activity)ctx).finish();
```

```
                ctx.startActivity(intent);
            }
            else{
                ((Register)ctx).enableButtons();
                progress.setVisibility(View.GONE);
                errormsg.setText(ctx.getResources().getString(R.string.userExist));
                errormsg.setVisibility(View.VISIBLE);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
},
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.e("Volley", "Error");
            }
        });
    RequestQueue rQueue = Volley.newRequestQueue(ctx);
    rQueue.add(jsObjRequest);
}
public void sendLoc(final String startPoint, final String endPoint, String ourtime, final Context ctx){
    data=new LocationData(ctx);
    String
url=serverIP+"car?startpos="+startPoint+"&finalpos="+endPoint+"&ourtime="+ourtime+"&id="+data.id()
;
    Log.d("Send mesage",url);
    JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
            public void onResponse(JSONObject response) {
                try {
                    ((GpsStatistics)ctx).setMidpoint(endPoint);
                    String distance =response.getString("distance");
```

```
                String time =response.getString("time");
                double dis=Double.parseDouble(distance);
                double disPref=Double.parseDouble(data.getTotalDistance());
                int timeInt=Integer.parseInt(time);
                data.total(dis+disPref+"",timeInt+data.getTotalTime());
            } catch (JSONException e) {
                e.printStackTrace();
            }
          }
        },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                   Log.e("Volley", "Error");
                }
            });
    RequestQueue rQueue = Volley.newRequestQueue(ctx);
    rQueue.add(jsObjRequest);
  }
  public void sendDestination(final String startPoint, final String endPoint,String username, final Context
ctx){
    data= new LocationData(ctx);
    String
url=serverIP+"destination?startpos="+startPoint+"&finalpos="+endPoint+"&username="+username ;
    Log.d("Send mesage",url);
    JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
           public void onResponse(JSONObject response) {
            try {
                String result =response.getString("Destination");
                data.id(result);
                Intent intent = new Intent(((Activity)ctx),StatTracking.class);
                ((Activity)ctx).finish();
                ctx.startActivity(intent);
```

```
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        Log.e("Volley", "Error");
                    }
                });
    RequestQueue rQueue = Volley.newRequestQueue(ctx);
    rQueue.add(jsObjRequest);
}
public void getDistanceOnRoad(double latitude, double longitude,
                        double prelatitute, double prelongitude, final Context ctx) {
    data=new LocationData(ctx);
    String startPoint=latitude+","+longitude;
    String endpoint=prelatitute+","+prelongitude;
    String url=serverIP+"distance?startpos="+startPoint+"&finalpos="+endpoint ;
    Log.d("Send mesage",url);
    JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
            public void onResponse(JSONObject response) {
                try {
                    String result =response.getString("distance");
                    Log.d("DistanceTest",result);
                    data.distance(result);
                    ((GpsStatistics)ctx).setDistance(result);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        },
```

```
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    Log.e("Volley", "Error");
                }
            });
    RequestQueue rQueue = Volley.newRequestQueue(ctx);
    rQueue.add(jsObjRequest);
}
public void findDestination(final Context ctx) {
    final String url=serverIP+"alldestination" ;
    Log.d("Send mesage",url);
    final JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
            public void onResponse(JSONObject response) {
                for(int i=1;i<=response.length();i++){
                    try {
                        JSONArray dest1=response.getJSONArray(i+"");
                        JSONObject dest2=dest1.getJSONObject(0);
                        fromList.add(dest2.getString("from"));
                        toList.add(dest2.getString("to"));
                        dest.add(i+". From: "+ fromList.get(i-1)+ "\nTo: "+ toList.get(i-1));
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
                destination=new Destinations();
                destination.Destinations(dest);
                Intent intent = new Intent(((Activity)ctx),FastDestination.class);
                ((Activity)ctx).finish();
                ctx.startActivity(intent);
            }
        },
            new Response.ErrorListener() {
```

```
            @Override
            public void onErrorResponse(VolleyError error) {
              AlertDialog.Builder builder;
              if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                builder = new AlertDialog.Builder(ctx, android.R.style.Theme_Material_Dialog_Alert);
              } else {
                builder = new AlertDialog.Builder(ctx);
              }
              builder.setTitle("Server Error")
                  .setMessage("Something goes wrong with the Server.Try Again.")
                  .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                      // continue with delete
                      ((SelectMap)ctx).enableButtons();
                    }
                  })
                  .setIcon(android.R.drawable.ic_dialog_alert)
                  .show();
            }
          });

    RequestQueue rQueue = Volley.newRequestQueue(ctx);
    rQueue.add(jsObjRequest);
  }
  public void getDirections(String time, String day, String id, final Context ctx) {
    String url=serverIP+"FastestRoute?time="+time+"&day="+day+"&id="+id;
    Log.d("Send mesage",url);
    JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
          public void onResponse(JSONObject response) {
            if(response.has("error")){
              AlertDialog.Builder builder;
              if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                builder = new AlertDialog.Builder(ctx, android.R.style.Theme_Material_Dialog_Alert);
```

```
                } else {
                    builder = new AlertDialog.Builder(ctx);
                }
                builder.setTitle("Route Error")
                        .setMessage("No Data found in Database for these route.Do you want to select new data for your route?")
                        .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int which) {
                                // continue with delete
                            }
                        })
                        .setNegativeButton(android.R.string.no, new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int which) {
                                // do nothing
                                Intent intent = new Intent(((Activity)ctx),MainActivity.class);
                                ((Activity)ctx).finish();
                                ctx.startActivity(intent);
                            }
                        })
                        .setIcon(android.R.drawable.ic_dialog_alert)
                        .show();
            }
            else{
                destination=new Destinations();
                destination.Directions(response);

                Intent intent = new Intent(((Activity)ctx),FastestWayMap.class);
                ((Activity)ctx).finish();
                ctx.startActivity(intent);
            }
        }
    },
            new Response.ErrorListener() {
                @Override
```

```
            public void onErrorResponse(VolleyError error) {
              AlertDialog.Builder builder;
              if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                builder = new AlertDialog.Builder(ctx, android.R.style.Theme_Material_Dialog_Alert);
              } else {
                builder = new AlertDialog.Builder(ctx);
              }
              builder.setTitle("Server Error")
                    .setMessage("Something goes wrong with the Server.Try Again.")
                    .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
                      public void onClick(DialogInterface dialog, int which) {
                        ((FastDestination)ctx).enableButtons();
                      }
                    })
                    .setIcon(android.R.drawable.ic_dialog_alert)
                    .show();
              Log.e("Volley", "Error");
            }
          });
    RequestQueue rQueue = Volley.newRequestQueue(ctx);
    rQueue.add(jsObjRequest);
}
public void isNetworkAvailable(final Context ctx) {
    ConnectivityManager connectivityManager
          = (ConnectivityManager) ctx.getSystemService(ctx.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
    if(activeNetworkInfo==null || !(activeNetworkInfo.isConnected()) ) {

      new AlertDialog.Builder(ctx)
            .setTitle(ctx.getString(R.string.connection))
            .setMessage(ctx.getString(R.string.noconnection))
            .setPositiveButton(ctx.getString(R.string.retry), new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int which) {
                isNetworkAvailable(ctx);
```

```
            }
        })
        .setNegativeButton(ctx.getString(R.string.exit), new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                System.exit(1);
            }
        })
        .setCancelable(false)
        .setIcon(android.R.drawable.ic_dialog_alert)
        .show();
    }
  }
}
```

2. FastWay Server

2.1. DBActions

```
public class DBActions {

	public String dbClass = "com.mysql.jdbc.Driver";
	private String dbName = "trafficapp";
	public String dbUrl = "jdbc:mysql://localhost:3306/" + dbName;
	public String dbUser = "root";
	public String dbPwd = "";
	private String url =
"https://maps.googleapis.com/maps/api/directions/json?origin=";
	private String url2 = "&destination=";
	private String key = "&key=personal-key ";
	private String url3 =
"http://maps.google.com/maps/api/geocode/json?address=";
	private ArrayList<String> waypoints;
	private ArrayList<String> duration;
	private ArrayList<String> distance;
	private ArrayList<String> sortedwaypoints;
	private ArrayList<String> sortedduration;
	private ArrayList<String> sorteddistance;
	private ArrayList<String> route;
	private ArrayList<String> coordinates;
	/**
	 * Method to create DB Connection
	 *
	 * @return
	 * @throws Exception
	 */
	public Connection DBConnention() throws Exception {
		Connection con = null;
		try {
			Class.forName(dbClass);
			con = DriverManager.getConnection(dbUrl, dbUser, dbPwd);
		} catch (Exception e) {
			// TODO
		}
		return con;
	}
	/**
	 * Method to check whether uname and pwd combination are correct
	 *
	 * @param uname
	 * @param pwd
	 * @return
	 * @throws Exception
	 */
	public boolean checkLogin(String uname, String pwd) throws Exception {
		boolean isUserAvailable = false;
		Connection dbConn = null;
		try {
			try {
				dbConn = DBConnention();
			} catch (Exception e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
```

```
                }
                Statement stmt = dbConn.createStatement();
                String query = "SELECT * FROM account WHERE username = '" +
uname + "' AND password=" + "'" + pwd + "'";
                ResultSet rs = stmt.executeQuery(query);
                while (rs.next()) {
                        isUserAvailable = true;
                }
        } catch (SQLException sqle) {
                throw sqle;
        } catch (Exception e) {
                // TODO Auto-generated catch block
                if (dbConn != null) {
                        dbConn.close();
                }
                throw e;
        } finally {
                if (dbConn != null) {
                        dbConn.close();
                }
        }
        return isUserAvailable;
}

/**
 * Method to insert uname and pwd in DB
 *
 * @param name
 * @param uname
 * @param pwd
 * @return
 * @throws SQLException
 * @throws Exception
 */
public boolean newUser(String uname, String pwd) throws SQLException,
Exception {
        boolean insertStatus = false;
        Connection dbConn = null;
        try {
                try {
                        dbConn = DBConnention();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                Statement stmt = dbConn.createStatement();
                String query = "SELECT * FROM account WHERE username = '" +
uname + "'";
                ResultSet rs = stmt.executeQuery(query);
                if (rs.next()) {
                        insertStatus = false;
                } else {

                        query = "INSERT into account(username, password)
values('" + uname + "','" + pwd + "')";
                        int records = stmt.executeUpdate(query);
                        // When record is successfully inserted
                        if (records > 0) {
```

```
                                insertStatus = true;
                        }
                }

        } catch (SQLException sqle) {
                // sqle.printStackTrace();
                throw sqle;
        } catch (Exception e) {
                // e.printStackTrace();
                // TODO Auto-generated catch block
                if (dbConn != null) {
                        dbConn.close();
                }
                throw e;
        } finally {
                if (dbConn != null) {
                        dbConn.close();
                }
        }
        return insertStatus;
    }
//Car Stats Insert
    public boolean insertCarData(String start, String finalpos, String ourTime,
String duration, String time,
                String dayofweek, String distance, String id) throws
SQLException, Exception {
        boolean insertStatus = false;
        Connection dbConn = null;
        try {
                try {
                        dbConn = DBConnention();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                Statement stmt = dbConn.createStatement();
                String query = "INSERT into
carstats(startpos,finalpos,idealtime,ourtime,daytime,day,distance,id) values('"
                                + start + "'," + "'" + finalpos + "','" +
duration + "','" + ourTime + "','" + time + "','"
                                + dayofweek + "','" + distance + "','" + id +
"')";
                System.out.println(query);
                int records = stmt.executeUpdate(query);
                // When record is successfully inserted
                if (records > 0) {
                        insertStatus = true;
                }
        } catch (SQLException sqle) {
                // sqle.printStackTrace();
                throw sqle;
        } catch (Exception e) {
                // e.printStackTrace();
                // TODO Auto-generated catch block
                if (dbConn != null) {
                        dbConn.close();
                }
                throw e;
```

```
		} finally {
			if (dbConn != null) {
				dbConn.close();
			}
		}
		return insertStatus;
	}
	public JSONObject returnDestinations() throws SQLException, Exception {
		boolean insertStatus = false;
		Connection dbConn = null;
		JSONObject destinationObject = new JSONObject();
		int i = 1;
		try {
			try {
				dbConn = DBConnention();
			} catch (Exception e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			Statement stmt = dbConn.createStatement();
			String query = "SELECT * FROM destinations";
			ResultSet rs = stmt.executeQuery(query);
			while (rs.next()) {
				JSONObject dest = new JSONObject();
				dest.put("from", namePlace(rs.getString("startpos")));
				dest.put("to", namePlace(rs.getString("finalpos")));
				JSONArray destArray = new JSONArray();
				destArray.put(dest);
				destinationObject.put(rs.getString("id"), destArray);

			}
		} catch (SQLException sqle) {
			// sqle.printStackTrace();
			throw sqle;
		} catch (Exception e) {
			// e.printStackTrace();
			// TODO Auto-generated catch block
			if (dbConn != null) {
				dbConn.close();
			}
			throw e;
		} finally {
			if (dbConn != null) {
				dbConn.close();
			}
		}
		return destinationObject;
	}
	public int insertDestination(String start, String finalpos,String username)
throws SQLException, Exception {
		boolean destinationexist = true;
		int i = 1;
		Connection dbConn = null;
		try {
			try {
				dbConn = DBConnention();
			} catch (Exception e) {
				// TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                }
                Statement stmt = dbConn.createStatement();
                ResultSet rs;
                rs = stmt.executeQuery("SELECT id FROM destinations");
                // System.out.println(rs.next());

                while (rs.next()) {

                        i++;
                }
                rs = stmt.executeQuery("SELECT * FROM destinations WHERE
startpos ='" + start + "'");
                // System.out.println(rs.next());
                while (rs.next()) {

                        if (rs.getString("finalpos").equals(finalpos)) {
                                //System.out.println(rs.getInt("id"));
                                i = rs.getInt("id");
                                destinationexist = false;
                                break;
                        }

                }
                if (destinationexist) {
                        int records = stmt.executeUpdate("INSERT into
destinations(startpos, finalpos,id) values('" + start
                                        + "','" + finalpos + "','" + i + "')");
                }
                destinationexist=true;
                rs = stmt.executeQuery("SELECT * FROM userdestination WHERE
startpos ='" + start + "' AND username='"+username+ "'");
                // System.out.println(rs.next());

                while (rs.next()) {

                        if (rs.getString("finalpos").equals(finalpos)) {
                                destinationexist = false;
                                break;
                        }

                }
                if (destinationexist) {
                        int records = stmt.executeUpdate("INSERT into
userdestination(startpos, finalpos,username) values('" + start
                                        + "','" + finalpos + "','" + username +
"')");
                }

        } catch (SQLException sqle) {
                // sqle.printStackTrace();
                throw sqle;
        } catch (Exception e) {
                // e.printStackTrace();
                // TODO Auto-generated catch block
                if (dbConn != null) {
                        dbConn.close();
                }
```

```
			throw e;
		} finally {
			if (dbConn != null) {
				dbConn.close();
			}
		}
		return i;
	}
	public boolean finishDistance(String startpoint, String finishpoint) {
		URLConnectionReader con = new URLConnectionReader();
		JSONArray part2;
		JSONObject part3;
		JSONArray part4;
		JSONObject part5;
		JSONObject distpart1;
		String distance;
		int dist;
		String finalurl = url + startpoint + url2 + finishpoint + key;
		try {
			JSONObject part = con.sendGet(finalurl);
			part2 = part.getJSONArray("routes");
			part3 = part2.getJSONObject(0);
			part4 = part3.getJSONArray("legs");
			part5 = part4.getJSONObject(0);
			distpart1 = part5.getJSONObject("distance");
			distance = distpart1.getString("text");
			String[] distanceArray = distance.split(" ");
			if(distanceArray[1].equals("m")){
				distance="0.0"+distanceArray[0];
			}

			else{
				distance = distanceArray[0];
			}
			dist = Integer.parseInt(distance);
			if (dist <= 0.5) {
				return true;
			} else {
				return false;
			}

		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		return false;
	}
	public String namePlace(String point) {
		URLConnectionReader con = new URLConnectionReader();
		JSONArray part2;
		JSONObject part3;
		JSONArray part4;
		JSONObject part5;
		String distpart1 = null;
		String distance;
		int dist;
		String finalurl = url3 + point;
```

```
			try {
				JSONObject part = con.sendGet(finalurl);
				System.out.println(finalurl);
				part2 = part.getJSONArray("results");
				System.out.println(part);
				part3 = part2.getJSONObject(0);
				distpart1 = part3.getString("formatted_address");


			} catch (Exception e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			return distpart1;
		}
//FastWay Algorithm!!!
		public ArrayList<String> selectRoute(String startPos, String finalPos,
String time, String day, String id)
				throws SQLException, Exception {

			route = new ArrayList<>();
			Random rand = new Random();
			int max=2;
			int min=0;
			int randomNum = rand.nextInt((max - min) + 1) + min;
			String point = startPos;
			Connection dbConn = null;
			int i = 0;
			dbConn = DBConnention();
			Statement stmt = dbConn.createStatement();
			while (true) {
				try {
					waypoints = new ArrayList<>();
					duration = new ArrayList<>();
					distance = new ArrayList<>();
					sortedwaypoints = new ArrayList<>();
					sorteddduration = new ArrayList<>();
					sorteddistance = new ArrayList<>();
					String query = "SELECT * FROM carstats WHERE id='" + id
+ "'AND daytime='" + time + "' AND day='" + day
							+ "' AND startpos='" + point + "'";
					System.out.println("Sql = " + query);
					ResultSet rs = stmt.executeQuery(query);
					while (rs.next()) {
						waypoints.add(rs.getString("finalpos"));
						duration.add(rs.getString("ourtime"));
						distance.add(rs.getString("distance"));
					}
					if(waypoints.size()==0){
						break;
					}

					findMax(waypoints.size());
					if(sortedwaypoints.size()>randomNum){
						route.add(sortedwaypoints.get(randomNum));
						point=sortedwaypoints.get(randomNum).toString();
					}
					else if(sortedwaypoints.size()>randomNum-1){
```

```
                                route.add(sortedwaypoints.get(randomNum-1));
                                point=sortedwaypoints.get(randomNum-
1).toString();
                        }
                        else{
                                route.add(sortedwaypoints.get(0));
                                point=sortedwaypoints.get(0).toString();
                        }
                        if(startPos.equals(point)){
                                return null;
                        }
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

                if (dbConn != null) {
                        dbConn.close();
                }
                return route;

        }
        public void findMax(int no) {
                int k = 1;
                while (k <= no) {
                        double max = Double.parseDouble(distance.get(0).toString());
                        int pos = 0;

                        for (int i = 1; i < waypoints.size(); i++) {
                                if ( Double.parseDouble(distance.get(i).toString()) >
max) {
                                max = Double.parseDouble(distance.get(i).toString());
                                        pos = i;
                                }
                        }
                        sortedwaypoints.add(waypoints.get(pos));
                        waypoints.remove(pos);
                        sortedduration.add(duration.get(pos));
                        duration.remove(pos);
                        sorteddistance.add(distance.get(pos));
                        distance.remove(pos);
                        k++;
                }
        }
        public ArrayList<String> findCor(String id) throws Exception {
                coordinates=new ArrayList<>();
                Connection dbConn = null;
                try {
                        try {
                                dbConn = DBConnention();
                        } catch (Exception e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                        Statement stmt = dbConn.createStatement();
```

```
                String query = "SELECT * FROM destinations WHERE id = '" + id
+ "'";
                ResultSet rs = stmt.executeQuery(query);
                if (rs.next()) {
                        coordinates.add(rs.getString("startpos"));
                        coordinates.add(rs.getString("finalpos"));
                }
        } catch (SQLException sqle) {
                throw sqle;
        } catch (Exception e) {
                // TODO Auto-generated catch block
                if (dbConn != null) {
                        dbConn.close();
                }
                throw e;
        } finally {
                if (dbConn != null) {
                        dbConn.close();
                }
        }
        return coordinates;
    }
    public void updateStats(String time, String distance,String username)
throws SQLException, Exception {
        boolean insertStatus = false;
        Connection dbConn = null;
        try {
                try {
                        dbConn = DBConnention();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                Statement stmt = dbConn.createStatement();
                String query = "SELECT * FROM userstats WHERE username = '" +
username + "'";
                ResultSet rs = stmt.executeQuery(query);
                if (rs.next()) {
                        String timeDb=rs.getString("totalTime");
                        String distanceDB=rs.getString("totalDistance");
                time=Integer.parseInt(time)+Integer.parseInt(timeDb)+"";
        distance=Integer.parseInt(distance)+Integer.parseInt(distanceDB)+"";
                        query="UPDATE userstats SET totalTime = '"+time+"',
totaldistance= '"+distance+"' WHERE username = '"+ username+ "'" ;
                        int records = stmt.executeUpdate(query);
                        // System.out.println(records);
                        // When record is successfully inserted
                        if (records > 0) {
                                insertStatus = true;
                        }
                } else {
                        query = "INSERT into userstats(username,
totalTime,totalDistance) values('" + username + "','" + time +
"','"+distance+"')";
                        int records = stmt.executeUpdate(query);
                        // System.out.println(records);
                        // When record is successfully inserted
                        if (records > 0) {
```

```
                                insertStatus = true;
                        }
                }
        } catch (SQLException sqle) {
                // sqle.printStackTrace();
                throw sqle;
        } catch (Exception e) {
                // e.printStackTrace();
                // TODO Auto-generated catch block
                if (dbConn != null) {
                        dbConn.close();
                }
                throw e;
        } finally {
                if (dbConn != null) {
                        dbConn.close();
                }
        }
}
        public JSONObject returnStats(String username) throws SQLException,
Exception {
                Connection dbConn = null;
                JSONObject destinationObject = new JSONObject();
                JSONObject finalObject =new JSONObject();
                int id=0;
                try {
                        try {
                                dbConn = DBConnention();
                        } catch (Exception e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                        Statement stmt = dbConn.createStatement();
                        String query = "SELECT * FROM userstats WHERE username = '" +
username + "'";
                        // System.out.println(query);
                        ResultSet rs = stmt.executeQuery(query);
                        if (rs.next()) {
                                finalObject.put("Time",rs.getString("totalTime"));

        finalObject.put("Distance",rs.getString("totalDistance")+ " klm");
                        }
                        else{
                                finalObject.put("Time",0);
                                finalObject.put("Distance",0+ " klm");
                        }
                         query = "SELECT * FROM userdestination WHERE
username='"+username+"'";
                        // System.out.println(query);
                        rs = stmt.executeQuery(query);
                        while (rs.next()) {
                                id++;
                                JSONObject dest = new JSONObject();
                                dest.put("from", namePlace(rs.getString("startpos")));
                                dest.put("to", namePlace(rs.getString("finalpos")));

                                JSONArray destArray = new JSONArray();
                                destArray.put(dest);
```

```
                        destinationObject.put(id+"", destArray);

                }
                if(id==0){
                        finalObject.put("Destinations","No Destinations");
                }
                else{
                        finalObject.put("Destinations",destinationObject);
                }
        } catch (SQLException sqle) {
                // sqle.printStackTrace();
                throw sqle;
        } catch (Exception e) {
                // e.printStackTrace();
                // TODO Auto-generated catch block
                if (dbConn != null) {
                        dbConn.close();
                }
                throw e;
        } finally {
                if (dbConn != null) {
                        dbConn.close();
                }
        }
        return finalObject;
    }
}
```

Βιβλιογραφία

[1] http://www.androidauthority.com/history-android-os-name-789433/

[2] https://www.android.com/versions/nougat-7-0/

[3] https://www.techlila.com/android-versions-with-list-names/#1.5

[4] https://www.android.com/versions/oreo-8-0/

[5] https://developer.android.com/about/dashboards/index.html

[6] http://blog.appversal.com/android-over-ios/

[7] https://www.tutorialspoint.com/android/android_architecture.html

[8] http://www.differencebetween.net/technology/internet/difference-between-api-and-web-service/

[9] https://www.guru99.com/web-service-architecture.html

[10] https://techterms.com/definition/web_service

[11] http://searchmicroservices.techtarget.com/definition/Web-services-application-services

[12] http://artemis.cslab.ntua.gr/el_thesis/artemis.ntua.ece/DT2014-0355/DT2014-0355.pdf

[13] https://www.w3.org/TR/ws-arch/

[14] https://www.ibm.com/developerworks/webservices/standards/

[15] http://www.json.org

[16] https://developer.android.com/studio/projects/index.html